Digitized by the Internet Archive
in 2013

http://archive.org/details/walshstoreapplic878brac

510.84
Il6r
no.878
Cop. 2

Math

Report No. UIUCDCS-R-77-878

UILU-ENG 77 1753

WALSHSTORE: THE APPLICATION OF BURST PROCESSING TO FAIL-SOFT STORAGE SYSTEMS USING WALSH TRANSFORMS

BY

EHUD BRACHA

October, 1977

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

JAN 18 1978

UIUCDCS-R-77-878

WALSHSTORE:
THE APPLICATION OF BURST PROCESSING TO FAIL-SOFT STORAGE SYSTEMS USING WALSH TRANSFORMS

BY

EHUD BRACHA

October, 1977

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1978

WALSHSTORE:
THE APPLICATION OF BURST PROCESSING TO FAIL-SOFT STORAGE SYSTEMS USING WALSH TRANSFORMS

Ehud Bracha, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1978

ABSTRACT

WALSHSTORE is concerned with the investigation into the application of orthogonal transforms, e.g., Walsh transforms, to the storage of visual images in a fail-soft manner, i.e., the system can sustain losses but the retrieved image can still be recognized, even if in a degraded form.

The application of Burst Processing to various parts in such a system is investigated together with an analysis of the amount of storage and speed of computation that can be achieved.

This paper presents the necessary theory and the implementation of the Walsh transformers and the fail-soft storage which were constructed and tested. Additionally, some software simulation results are compared to the actual machine performance.

ACKNOWLEDGEMENT

The author wishes to express his deep gratitude to his advisor, Professor W. J. Poppelbaum, for suggesting the topic for this thesis and for the help and encouragement, and to Professor J. Liu and Professor W. Kubitz for the helpful discussions.

Thanks are also due to the DCL electronic shop personnel, especially Frank Serio and Sam Macdowell, for their invaluable help in the construction and photographing of the WALSHSTORE machine; to Stan Zundo for the help in the drafting and to Cinda Robbins for the help in the typing.

Last, but not least, the author wishes to thank his wife, Yehudit, for her great help in the proof reading and all the art work involved in this thesis, for her infinite patience and understanding and for the encouragement at all those moments when nothing seemed to be going right.

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
2.	THEORY OF ORTHOGONAL TRANSFORMS	3
	2.1 General Theory	3
	2.2 The Choice of Walsh Transforms	9
	2.3 Storage System Organization	11
	2.4 Storage Size and Speed Analysis for Large Pictures	15
3.	APPLICATION OF BURST PROCESSING	20
	3.1 Image Pick-up and Display Using Burst Processing	21
	3.2 IS (RIS) Organization	23
	3.3 Transformer Design and Analysis	25
	3.4 The Application to Large Pictures	31
4.	MACHINE DESCRIPTION AND RESULTS	34
	4.1 Camera and Display Control	35
	4.2 Information Store	37
	4.3 Transformer	40
	4.4 Convolution Store	42
	4.5 Inverse Transformer	45
	4.6 Retransformed Information Store	49
	4.7 Control Unit	51
	4.8 Results	54
5.	CONCLUSIONS	59

Page

APPENDICES

A. PA's and PA-Trees..... 61

B. The CYCLOPS Digital Camera..... 63

C. Operating Instructions for WALSHSTORE..... 65

LIST OF REFERENCES..... 68

VITA..... 69

LIST OF FIGURES

Figure		Page
2.1	A Few Examples of Orthogonal Functions	5
2.2	Data Compression in the Transform Domain	8
2.3	WALSHSTORE System Block Diagram	12
2.4	Divisions of the Transform Domain into Quadrants	14
2.5	Division of a Picture into Subpictures	16
2.6	WALSHSTORE Block Diagram for Sequential Subpicture Processing	18
3.1	Information Store Organization	24
3.2	PA-Tree Diagram	27
3.3	Comparison of a Burst Transformer to a Binary Transformer	30
4.1	Camera and Display Control Block Diagram	36
4.2	Information Store Block Diagram	38
4.3	Transformer Block Diagram	41
4.4	Convolution Store Quadrant Block Diagram	43
4.5	Inverse Transformer Block Diagram	48
4.6	Retransformed Information Store Block Diagram	50
4.7	Control Unit Block Diagram	52
4.8	Software Simulations for the Pattern A	55
4.9	Results from WALSHSTORE for the Pattern A	58
A-1	Perverted Adder Diagram	62
B-1	The CYCLOPS Digital Camera Output Video Format	64
C-1	Photograph of WALSHSTORE with CYCLOPS and Display Units	66

1. INTRODUCTION

The purpose of WALSHSTORE, as initially proposed by Poppelbaum [5], is the research into and the design of a fail-soft storage system for black and white square pictures, e.g., maps or TV frames. The fail-soft property suggests that a portion of the storage may be destroyed but the stored picture can still be retrieved, even if with some degradations. The fail-softness in WALSHSTORE is achieved by using Walsh transforms, which produce an orthogonal transformation of the original image into a domain where different coefficients carry different amounts of information and, therefore, some are more "important" than others. This property is called the compression property and it is associated with all orthogonal transforms. WALSHSTORE represents an attempt to perform all the processing digitally and to implement the various system components using Burst Processing, where possible.

Burst Processing, the deterministic counter-part of stochastic processing, which has been used for the last three years in various applications in the IEL group in the University of Illinois [2,3,9-13], uses a unary representation where numbers are represented by the number of pulses which are ON ("1") during a frame, or a so-called block. A block length of 10 has been traditionally used. However, other block lengths can be used as well, e.g., 8 or 12. The precision of a block of information is basically limited to about 10 percent and more precision may be gained by averaging over many blocks. Clearly, Burst Processing is useful in those applications where a high precision is not necessary and where there is a saving in hardware due to the simplicity of the processing units.

For a block length of 12 the numbers to be represented range from 0 to 12. For signed numbers there are two basic methods that can be used. In the biased representation the range 0 to 12 is mapped into the range -6 to +6 and then 0 represents -6, 6 represents 0 and 12 represents +6. This representation is useful in applications where only additions and subtractions are used. In the sign and magnitude representation there is a special sign bit which is added to the regular 12 pulses Burst and then the Burst itself represents the magnitude and the sign bit represents the sign of the number being processed. This is useful where multiplications and divisions are mainly performed.

Burst Processing is adopted here for picture processing since most pictures can be recognized when displayed with a small number of grey levels, e.g., 10-16. Additionally, the processing part has proved to be comparable in complexity to that of a binary system.

Section 2 in this paper provides the mathematical background concerning the use of orthogonal transforms in picture compression, a discussion on the particular choice of the Walsh transform and also describes the system organization of WALSHSTORE. In section 3 the application of Burst Processing in the various parts of the system is discussed and a detailed description of these parts is given together with the analysis of the application to large pictures. Section 4 provides a detailed description, on the block diagram level, of the actual WALSHSTORE machine that was constructed and presents some results that portray the fail-soft properties of the machine. Finally, in section 5 conclusions are drawn and some notes about possible improvements are presented.

2. THEORY OF ORTHOGONAL TRANSFORMS

2.1 General Theory

The orthogonal transforms that this paper is concerned with are those that are essentially linear mappings, i.e., the coefficients in the transform domain are a linear combination of some of or all the points in the original domain. When the quantities measured in the original domain are sampled and quantized the transform may be discrete and digital processing may be used. In this case the sampled quantities can be represented by vectors in one-dimensinal transforms and by matrices in two-dimensional transforms. Examples are speech processing and picture processing, respectively. Throughout this paper it is assumed that discrete two-dimensional transforms are used to transform digitized two-dimensional pictures, i.e., the quantities being quantized and transformed are grey levels.

An orthogonal transform uses a set of orthogonal functions and these can be represented by a square matrix. Let [T] be such an NxN matrix. T_i, i= 0,1,...,N-1, are the rows of [T] and they are also the orthogonal functions that define the orthogonal transform. The orthogonality of [T] requires that:

$$T_i \cdot T_j = \begin{cases} c & i=j \\ 0 & i \neq j \end{cases} \quad \text{for all } i,j = 0,1,\ldots,N-1. \text{ (Vector dot Product)}$$

Additionally, for the transform to be useful, it is necessary that $[T]^{-1}$ exists in order to be able to retrieve the transformed image. Let [X] be the NxN matrix representing the sampled picture of NxN points, let [Y] be another NxN matrix representing the transformed picture and let [X'] be the NxN matrix representing the inverse transformed picture.

The discrete transform of [X] into [Y] can be written as the following matrix equation:

$$[Y] = [T] \cdot [X] \cdot [T] \tag{1}$$

Applying basic matrix multiplications and making use of $[T]^{-1}$, it can easily be proved that the inverse transform of [Y] into [X'] can be written as:

$$[X'] = [T]^{-1} \cdot [Y] \cdot [T]^{-1} \tag{2}$$

When [T] is a unitary transform $[T]^{-1} = \frac{1}{c}[T]^T$ and then,

$$[X'] = \frac{1}{c^2} \cdot [T]^T \cdot [Y] \cdot [T]^T = \frac{1}{C} \cdot [T]^T \cdot [Y] \cdot [T]^T \tag{3}$$

and the inverse transform involves essentially the same computations as those in the transform. When equations (1) and (3) are written in a sum notation the resulting equations are:

$$Y(k,m) = \sum_{1,n = 0,1,\ldots,N-1} X(1,n) \cdot T(k,1) \cdot T(m,n) \tag{4}$$

$$\text{for } k,m = 0,1,\ldots,N-1$$

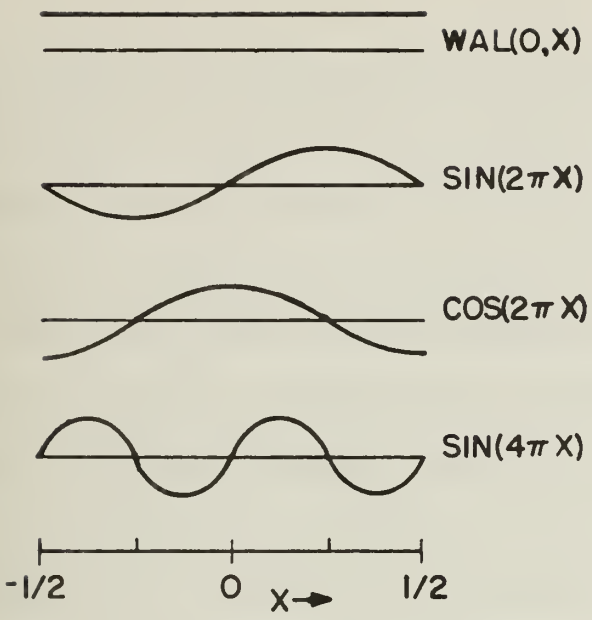
and

$$X'(1,n) = \frac{1}{C} \sum_{k,m = 0,1,\ldots,N-1} Y(k,m) \cdot T(k,1) \cdot T(m,n) \tag{5}$$

$$\text{for } 1,n = 0,1,\ldots,N-1$$

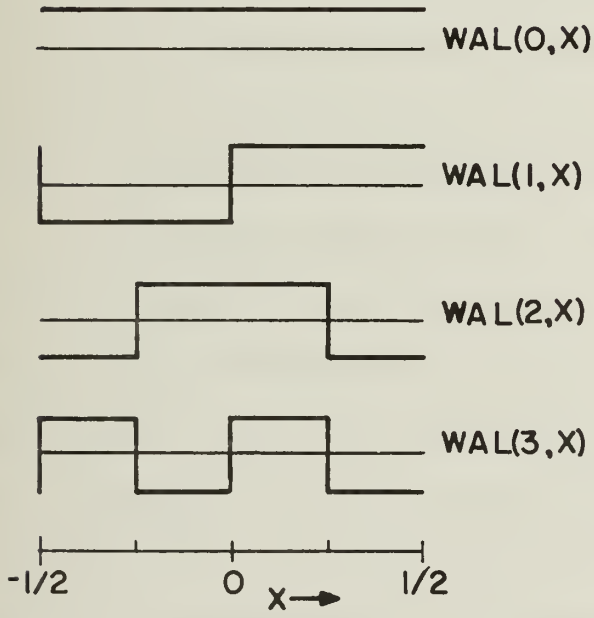
It should be noted that in the continuous case the summations are simply interchanged with integrations. (See [1-Ch. 2] for a discussion on both continuous and discrete transforms.)

A few examples of orthogonal functions are shown in Figure 2.1 and for each set of functions both the continuous and the discrete set is shown for N=4. The Sine-Cosine functions are used in the well-known Fourier transform and Walsh functions are used in Walsh transform. These are the most used transforms in signal processing applications.



$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -0.7 & -0.7 & 0.7 & 0.7 \\ -0.7 & 0.7 & 0.7 & -0.7 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

(a)



$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

(b)

Figure 2.1 A Few Examples of Orthogonal Functions

(a) Sine-Cosine

(b) Walsh

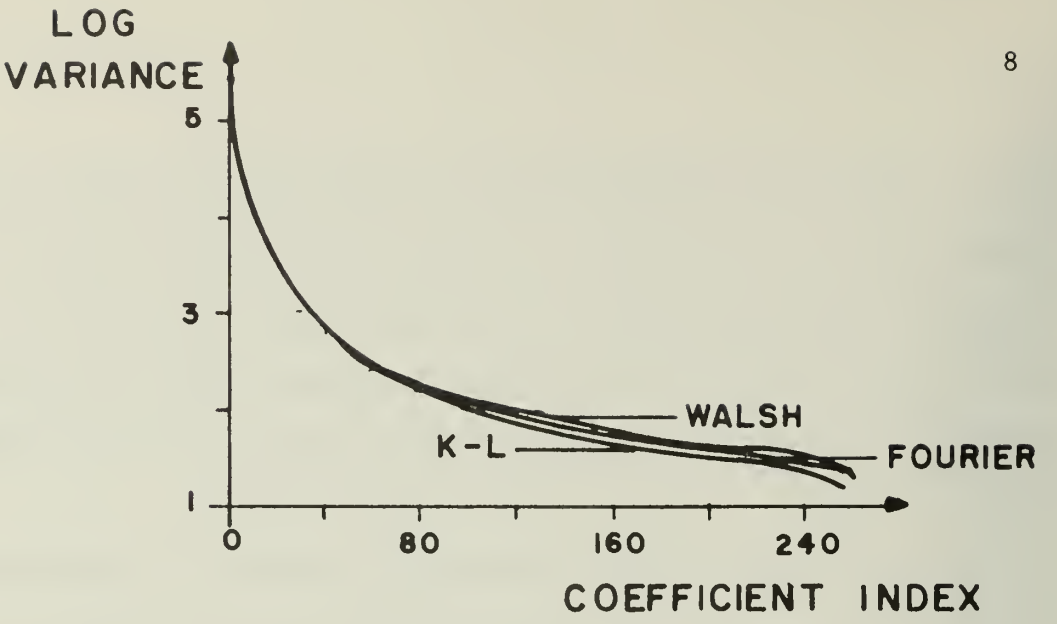
As can be observed from Figure 2.1, Walsh functions are two-valued whereas the Sine-Cosine functions are variable and, therefore, an encoding of the functions at discrete points requires more than one-bit words. However, the increased accuracy inherent in the Sine-Cosine functions results in transform errors in Fourier transforms which are smaller than those produced by Walsh transform when the same number of functions is used.

As mentioned in the introduction the orthogonal transform is used in order to achieve data compression. The desire is to represent a picture, in which there is a lot of correlation between adjacent points, with the minimum number of coefficients necessary to reconstruct the picture faithfully. This can be achieved if the correlation between the coefficients is zero or small. The only orthogonal transform that produces zero correlation between the transform coefficients is the the Karhunen-Loeve Transform (KLT). The orthogonal functions in this transform are not fixed like those in the Fourier or Walsh transforms, but they have to be computed from the statistical properties of the pictures (or set of pictures) being transformed. This process involves the solution of an eigenvector problem. [6-Ch. 9 and 1-Ch. 5]. Once such functions are computed for a given picture KLT will produce the minimal mean square error in the inverse transformed picture with respect to the original picture and there will be no correlation between the transform coefficients, i.e., KLT is optimal in the sense of the mean square error.

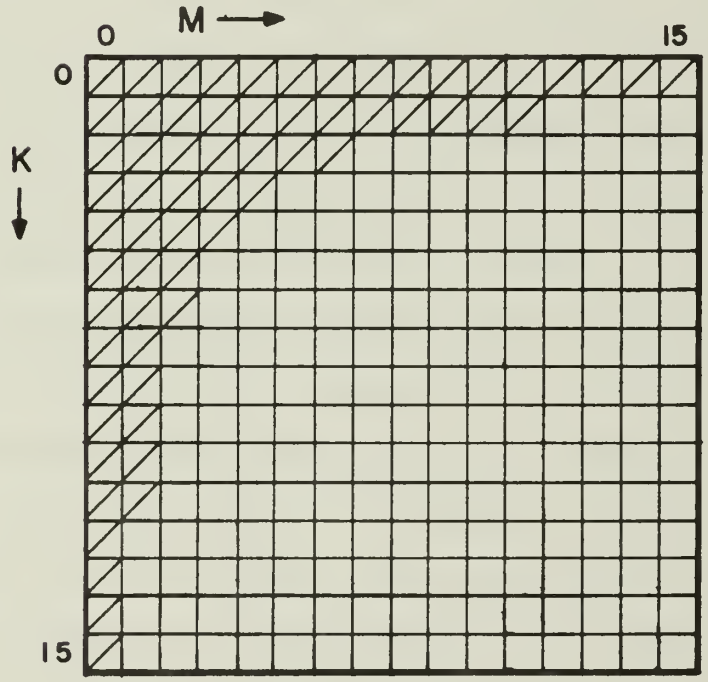
The main difficulty with the KLT is the fact that the functions are not known a priori and their computation is very complex. If the Fourier transform or the Walsh transform is used, the correlation between coefficients is not zero and they are not optimal in the mean square error

sense. However, the computation is a lot simpler and the error is not increased very much with respect to the error in the KLT, making the use of these transforms very attractive. [1-Ch. 5, 14]. Additionally, it should be noted that the KLT does not, in general, have a fast algorithm like those that Fourier and Walsh transforms have, and this fact makes them even more attractive, especially for a hardwired system.

All the three transforms discussed above possess the compression property. Moreover, the information tends to be concentrated in the low frequencies (or sequencies, as defined by Harmuth [7]), coefficients, i.e., they are the "important" coefficients. Figure 2.2 shows this property resulting from experiments in both one-dimensional and two-dimensional transforms. See [14] and [6-Ch. 9]. (The variance in Figure 2.2(a) represents the relative energy carried by the coefficients.) This points to a possible solution to the fail-soft storage in the transform domain. The high order coefficients may be lost without a great effect on the reconstructed picture but the low order coefficients are important and a means to protect them is necessary. A way to do this is to simply duplicate the important information. Another way can be to store important features in the picture in a different form so that when the low order coefficients are lost, this back-up storage can supply the details about the important features pertinent to the user. Obviously, the first choice is a lot simpler to implement in hardware even if it may be inferior in performance to the other, and it is this choice which is used in the actual machine that was constructed.



(a)



(b)

Figure 2.2 Data Compression in the Transform Domain

(a) One-Dimensional Case

(b) Two-Dimensional Case - The most important coefficients marked with diagonal lines

In the above approach all the coefficients are transformed and all are assigned the same amount of storage. Consequently, the probability of loss of a high order coefficient is equal or higher than that of the loss of a low order coefficient, since the latter is duplicated. Other possible approaches are discussed in the Conclusions section.

2.2 The Choice of Walsh Transforms

So far it has been shown that although the KLT is superior to Walsh and Fourier transforms, the latter are widely used because of their simplicity. The reasons why the Walsh transform has been chosen are the following: 1) The two-valued Walsh functions can be easily mapped into digital 0,1 signals and then digital processing may be readily used. 2) A product of two Walsh functions may be +1 or -1 only, i.e., the value being transformed is either added or subtracted and, consequently, no multiplications are required, resulting in much faster and a lot simpler processing. 3) Although the error and the correlation between coefficients are not optimal they differ from the optimal only slightly and thus they do not prohibit the use of the transform.

The equations (4) and (5) to be implemented become now:

$$Y(k,m) = \frac{1}{N} \sum_{1,n} \sum X(1,n) \cdot WAL(k,1) \cdot WAL(m,n) \qquad (6)$$

$$\text{for } k,m = 0,1,\ldots,N-1$$

and

$$X'(1,n) = \frac{1}{N} \sum_{k,m} \sum Y(k,m) \cdot WAL(k,1) \cdot WAL(m,n) \qquad (7)$$

$$\text{for } 1,n = 0,1,\ldots,N-1$$

where WAL(k,1) represents Walsh function k at position 1.

For an NxN picture the first N Walsh functions are required. The scaling factor from equation (5) is equal to N^2 since $[WAL]^{-1} = \frac{1}{N}[WAL]^T$, and it has been distributed between the transform and the inverse transform computations. This is a compromise between the two following cases: 1) Scaling by N^2 in the inverse transform, as originally proposed, which causes the transform coefficients word length to be too long. 2) Scaling by N^2 in the transform reduces the word length necessary for the coefficients, however, the scaling also produces too much error in the coefficients due to truncation. The distribution of C between the two computations is a compromise and extensive simulations have indicated that this choice is acceptable, i.e., the amount of error introduced by the scaling by N in the transform process produces such errors that even after the inverse transform the average error is less than one unit of intensity. (See [8] for their discussion on truncation in the transform.) It should be noted that all computations are performed with fixed point arithmetic.

The software simulations were used as an aid in various decisions, once it was decided to use Walsh transforms. The transform domain was divided into four stripes, each stripe containing exactly one fourth of the rows, and then a loss of one, two and three stripes (quadrants) was simulated. The conclusive results, as expected from theory, showed that the first quadrant was the most important and its loss was intolerable. This quadrant is the one that is duplicated. Some results from the simulations are shown later in this paper, in Figure 4.8.

2.3 Storage System Organization

Figure 2.3 shows the WALSHSTORE block diagram. The information store (IS) receives an image from the image sensor and then the image may be displayed on the IS display. The transformer (T) transformes the contents of IS into the convolution store (CS). This store is divided into four identical quadrants, as previously mentioned. The first quadrant is duplicated since it stores the most important part of the information. When retrieval of the image is required, the inverse transformer (IT) transformes the contents of CS into the retransformed information store (RIS). The result can then be displayed on the RIS display. The division of CS into five different parts is done so that they can be placed in different physical locations: Then their probabilities of failure are independent. The control unit accepts status information from the different quadrants and makes a decision about which one is used in the case there are duplicates, i.e., the first quadrant. It also supplies all the control signals to the various units during all modes of operation.

It should be noted that all units performing processing or control functions can be duplicated for fault tolerance.

The choice of the division of CS into four quadrants, each quadrant being a stripe of rows in the transform domain as shown in Figure 2.4(a), is somewhat arbitrary and other choices could be used. Recalling from Figure 2.2(b) that the important coefficients are, in most cases, those that are close to the k and m axes, another division can be used where quadrant one includes these important coefficients and then quadrants 2, 3 and 4 are divided as shown in Figure 2.4(b). This division may be called the "statistical" division since statistically the probability that quadrant

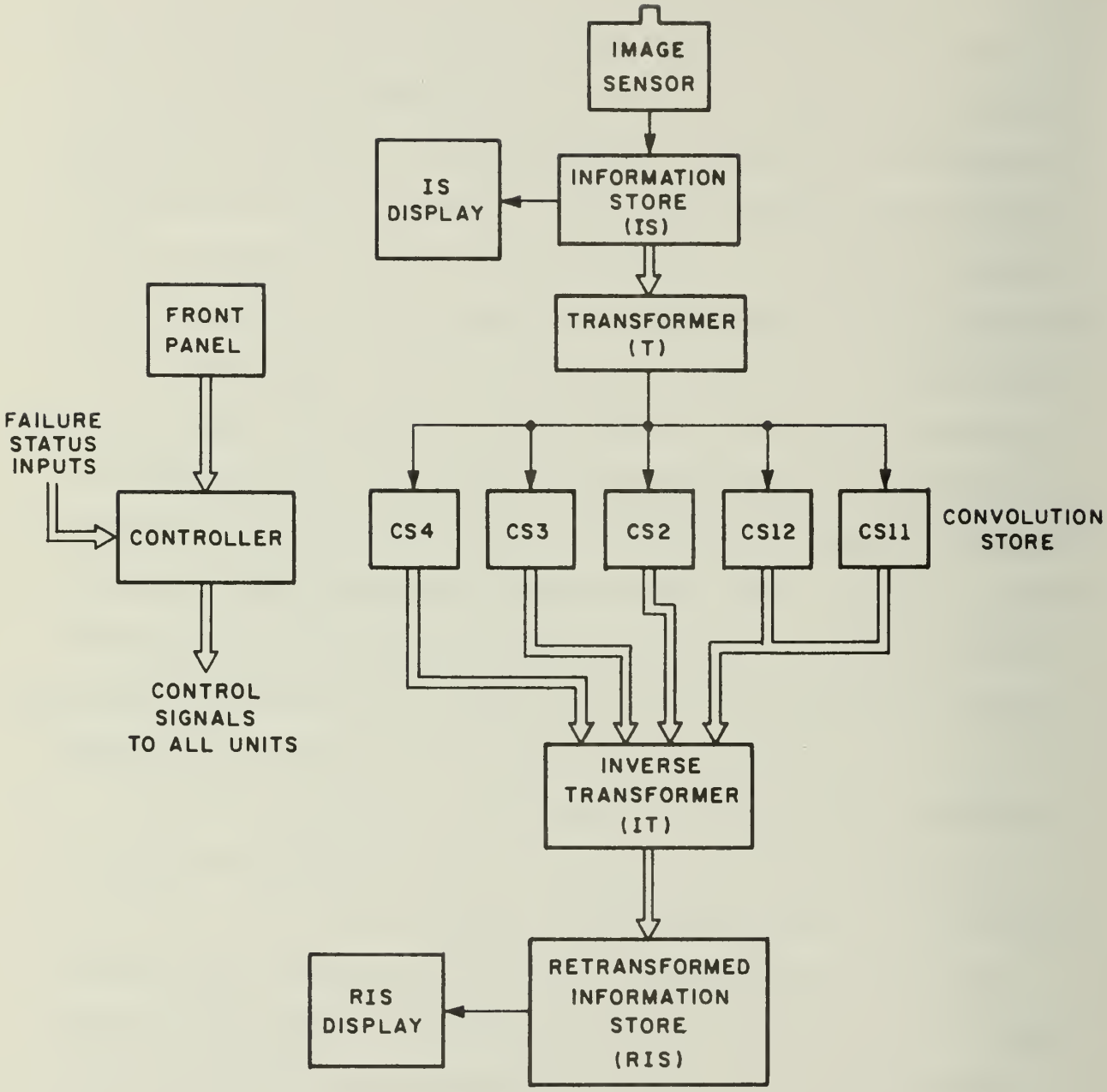
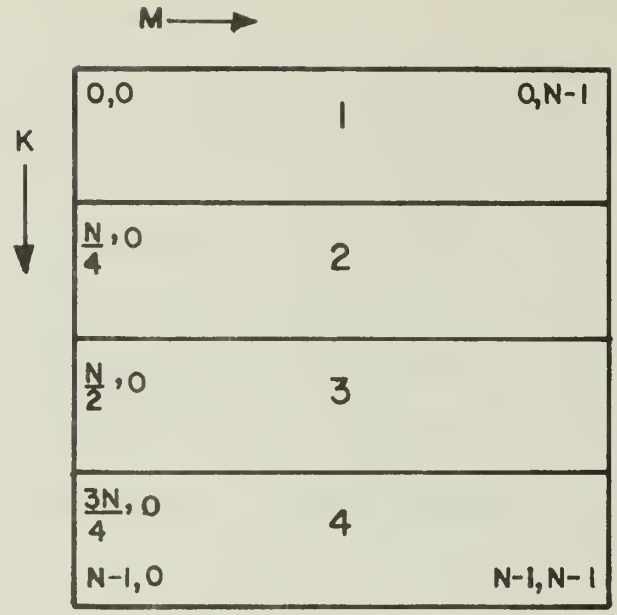


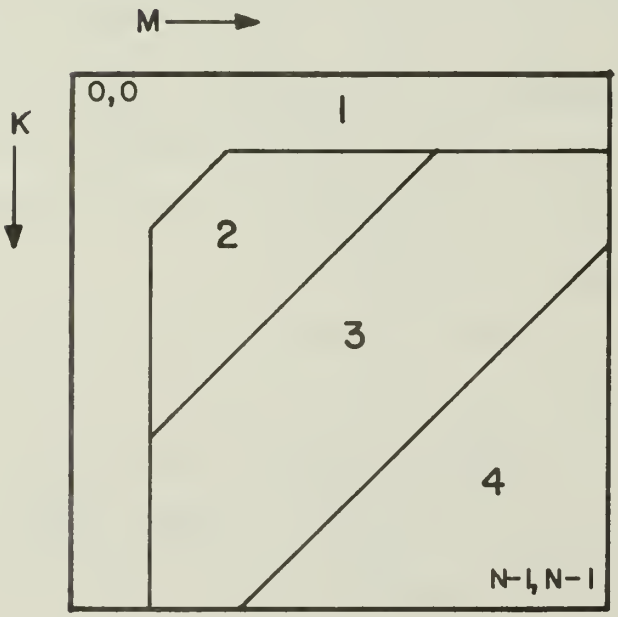
Figure 2.3 WALSHSTORE System Block Diagram

one includes the coefficients with the highest information content is very close to 1. This "statistical" division is better than the previous one in terms of compression of information into quadrant one. However, it has its difficulties. One difficulty is that different quadrants may have different number of coefficients making parallel processing impossible. Another difficulty is that the transformers must know into what quadrant each coefficient should be stored or read from, thus requiring a special address storage. In the stripes division these problems are solved automatically since all quadrants include the same number of coefficients and the index of each coefficient immediately points at the quadrant into which it is stored or from which it is read during the transforms. Moreover, as will be shown in section 4.5, the inverse transform may be easily performed in parallel since a complete column of coefficients is available at any one time. All the above simplifications have lead to the decision to use the stripes division and, as pointed out earlier, the simulations have indicated that this choice is right.

It should be noted that although pictures of 32x32 points only were used in the software simulation, this is not a claim that this picture size is necessarily the optimal. In the next section the use of subpictures is explained and a discussion on practical subpicture sizes is given together with the application to large pictures.



(a)



(b)

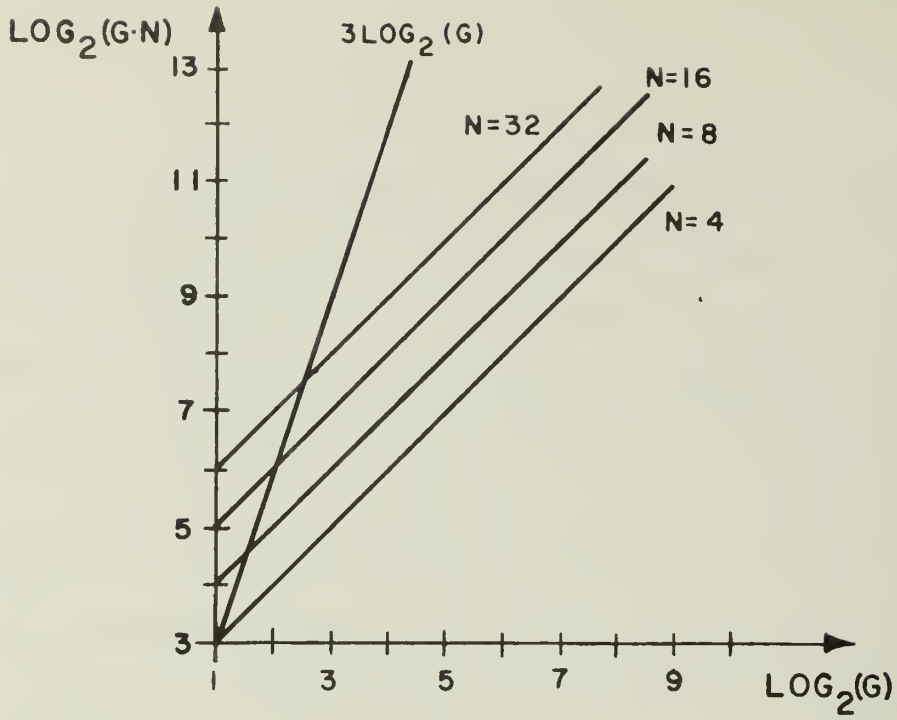
Figure 2.4 Divisions of the Transform Domain into Quadrants

(a) Stripes Division

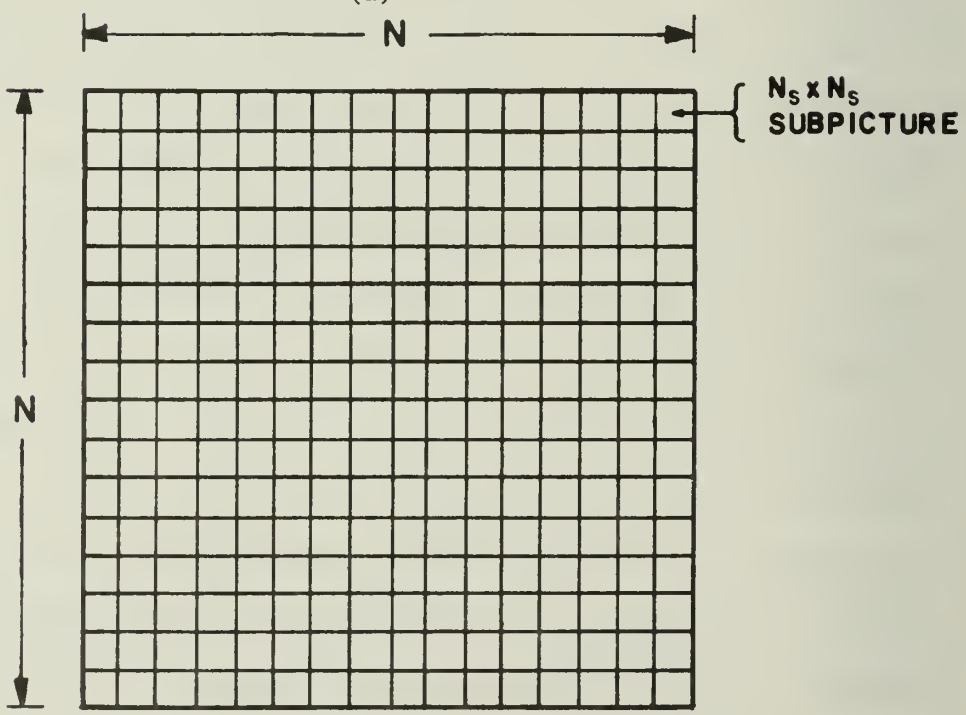
(b) "Statistical" Division

2.4 Storage Size and Speed Analysis for Large Pictures

Equations (6) and (7) in section 2.2 provide the basis for the analysis of the CS word size and the transform speed. Observing these equations, it is clear that the number of operations necessary in order to compute a complete transform is N^4, since there are N^2 Y coefficients and each requires N^2 additions or subtractions of the N^2 X(1,n)'s. The scaling can be done on the fly. Let G be the number of grey levels present in the input X; Clearly, $\log_2(G)$ bits are necessary to store such information. Since N^2 additions are involved in each coefficient computation the final CS word length is $\log_2(G \cdot N^2)$ bits before the scaling and $\log_2(G \cdot N)$ bits after it. For example, for N=32 the number of operations amounts to $2^{20} \simeq 10^6$ and then if 15 grey levels are used the CS word length is 9 bits, whereas the original word length is 4 bits. The ratio is smaller than 3 and remains so for all G's larger than 8 in the case where N=32, as shown in Figure 2.5(a). However, practical pictures are usually of 512x512 points in size, and then the number of operations jumps to $2^{36} \simeq 6.8 \times 10^{10}$. This figure is not practical, since even a clock signal of 100 Mhz gives a computation time of 6.8×10^2 sec. and the CS word length jumps to 13 bits so that the ratio to the original word length exceeds 3, suggesting that a simple triple modular redundant system (TMR) can be used since it requires less storage. In addition to these problems, the number of Walsh functions to be used in the transform is 512, meaning that many functions have such a high sequency that they represent information that cannot be recognized by the eye since the eye is limited in its frequency response.



(a)



(b)

Figure 2.5 Division of a Picture into Subpictures

In order to decrease the number of operations and to keep the CS word length low when transforming a large picture, it is possible to divide the picture into smaller subpictures, as shown in Figure 2.5(b), and then transform the subpictures separately. The size of the subpicture should not be too high for the reasons given before, and it should not be too small because all the coefficients will be of low sequency and, therefore, all will be important and not just a group of them. A subpicture size of 8 or 16 seems to be practical. (See [14]).

Let $N_s xN_s$ be the number of points in a subpicture. The word length of each CS coefficient, when an NxN picture is transformed as a whole, is $\log_2(G \cdot N)$ bits whereas when subpictures are transformed the length is $\log_2(G \cdot N_s)$ bits, thus giving a saving of $\log_2(N/N_s)$ bits for each coefficient. The number of operations for a whole picture transform is N^4 whereas when subpictures are transformed this number is only $N^2 \cdot N_s^2$ since each subpicture requires N_s^4 operations and there are N^2/N_s^2 subpictures. This represents a speed up of N^2/N_s^2. For example, if N=512 and N_s=16 the saving in storage is 5 bits per coefficient, from 13 bits to 8 bits, and the number of operations is reduced by a factor of 1024, from 2^{36} to 2^{26}. Obviously, the savings are so great when subpictures are used that this approach should be preferred.

Figure 2.6 shows a possible system block diagram using $N_s xN_s$ subpictures and shift registers for storage. The subpictures are stored and processed sequentially in all the stores. In order to speed up the transform processes even more the following schemes can be used: 1) Use the Fast Walsh transform. 2) Use parallelism in the regular transform, i.e., simply add many points at one time. These schemes reduce the transform

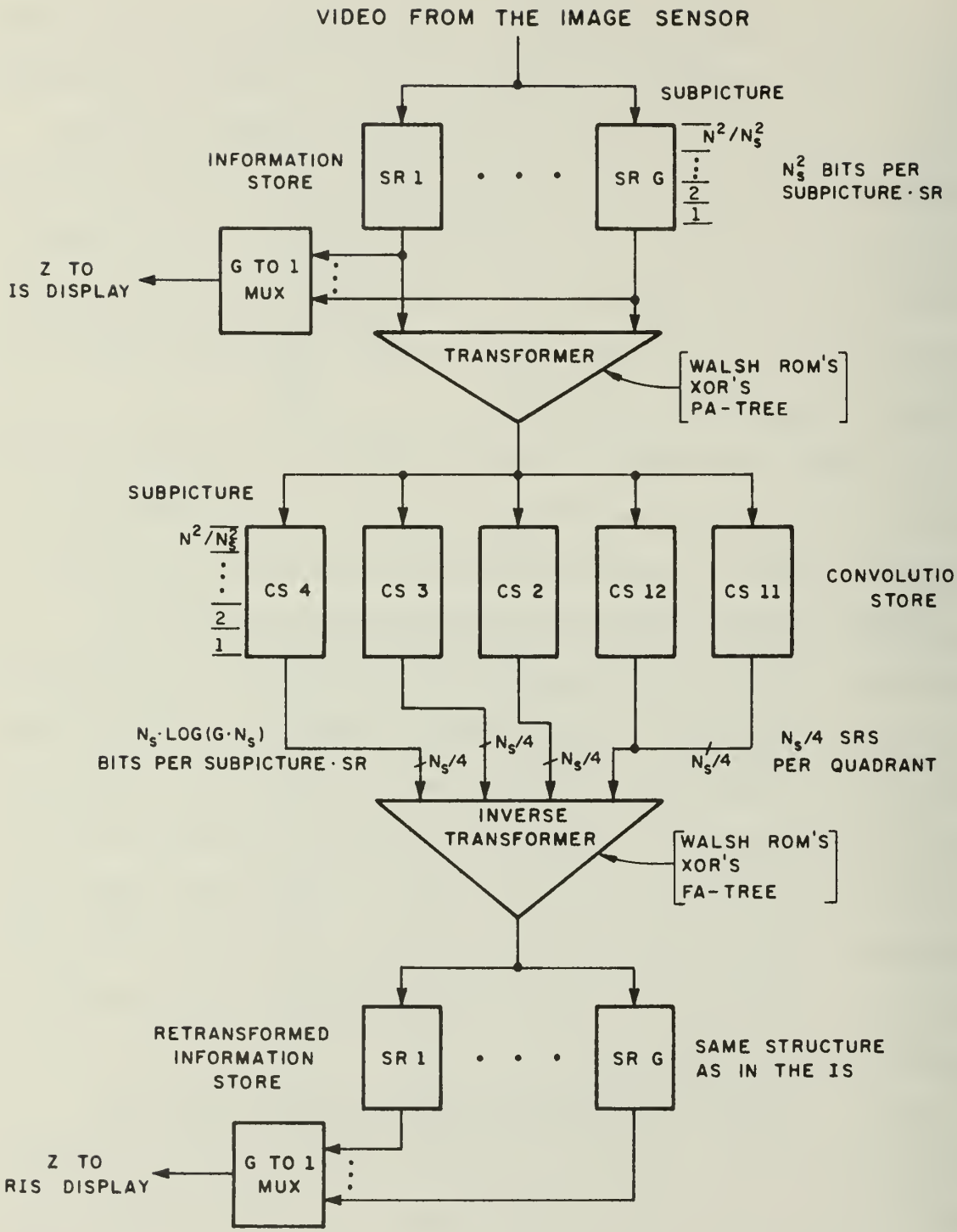


Figure 2.6 WALSHSTORE Block Diagram for Sequential Subpicture Processing

computation period for each subpicture. On the complete picture level it is possible to use an array of processors, each processor dedicated to one subpicture or a group of subpictures and all processors can work in parallel to achieve a very high speed. The usual trade-off between speed and complexity can determine which approach to choose.

A few words about the absolute transform speed. If the regular transform is used with N=512 and N_s=16 the number of operations is $2^{26} \simeq 64 \times 10^6$ and a 100 Mhz clock signal results with a transform period of 0.64 sec. A speed up scheme, as described previously, can increase this speed by at least a factor of 4 and as much as 1024, so a suitable scheme can be chosen to fit the particular application.

3. APPLICATION OF BURST PROCESSING

Burst Processing, as the name implies, is an efficient way of processing digital, or digitally encoded analog, signals. In storage applications, however, the Burst representation is clearly inefficient since n bits of Burst are necessary in order to store the information contained in only $\log_2(n)$ bits when binary representation is used. Consequently, Burst representation in these applications should be used where the overall system complexity is reduced by the use of Burst techniques, e.g., simpler encoders and processing units, and where the ratio of $n/\log_2(n)$ does not exceed a limit, e.g., 2 to 3. The cost of such a storage is not $n/\log_2(n)$ times higher than the cost of a binary storage and, also, the cost reduction due to simpler processors or encoders may compensate for the additional cost of the storage. The overall benefit is in the simpler (and easier to control) processors or encoders, and this overshadows the need to increase the amount of storage.

In [2] it has been demonstrated that TV pictures can be encoded, transmitted and decoded with reasonable resolution using Burst techniques with only ten levels of intensity. Additionally, there exist image sensors that can provide Burst encoded pictures directly; as a result, the image sensor, IS and RIS in a WALSHSTORE type system can use Burst Processing. (Figure 2.3). Since T receives the information from IS it is natural for T to use Burst Processing. Also, as will be shown later in this section, the Burst T is comparable in complexity to a binary T. The remaining units to be considered are CS and IT. CS stores the transform coefficients generated by T and since the latter adds many numbers in order to produce

each coefficient, the CS word length is usually so high that the ratio $n/\log_2(n)$ becomes too high for Burst representation to be practical. CS, therefore, should use the binary representation, except in cases where the CS word length is small. (This is the case when a very small subpicture size is used, e.g., 2x2.) As for IT, it receives the information from CS and it is natural that it uses binary techniques. The use of Burst Processing here would require a conversion from binary to Burst and then in addition to the extra hardware this method would be very slow because, in general, $\log_2(n)$ time slots would have to be converted into n time slots during the processing.

The rest of this section consists of a description of the principles of operation of the units using Burst Processing, i.e., the image sensor, IS (and RIS) and T, and an analysis of the changes in the overall system organization when large pictures are processed.

3.1 Image Pick-up and Display Using Burst Processing

Image pick-up may be performed in various ways. In most of them the image is scanned horizontally line by line and when a whole picture is complete, the scan is started all over, and so on. The video information may be encoded into digital signals in many ways. The conventional way is to encode each sampled point into a (usually binary) word. One clock period is required, usually, to encode the intensity of each point. The CYCLOPS camera, described in detail in section 4.1 and in Appendix B, provides a Burst encoded output. This camera scans the image 15 times and during each scan (field) it produces only one unit of intensity for each sampled point. After the 15 fields are scanned each point has been sampled

exactly 15 times and the number of 1's for the point represent its intensity. The 16th field is used for clearing the image sensor. Clearly, the clock signal frequency for such a camera is 16 times higher than the one for the conventional method since the image is scanned in full 16 times instead of just once. On the other hand the complexity of the encoder is reduced from an ADC to, essentially, a comparator which decides about the output. However, if large pictures are picked up, it is obvious that a CYCLOPS type camera cannot be used in its current form. Section 3.4 provides a few solutions to this problem. It should also be mentioned that when the conventional way of image pick-up is used, a parallel Burst encoder is less complicated than the binary encoder (ADC) since the latter requires a bank of comparators in order to decide which level is being encoded and the outputs from these comparators, which form exactly a spatial Burst, are encoded into binary. The Burst encoder simply uses the comparators' outputs and avoids the added hardware and is, therefore, simpler.

Image display may be done in the same way it was picked up. If the conventional method is used, a DAC or a Burst to Analog converter is required so that for every point the digital information is converted into analog and then used to control the intensity of the display beam. If the serial Burst encoding method is used, the image is scanned 16 times and during each field each point is controlled by the appropriate unit of intensity corresponding to the point and the field. The display screen's integration property will average the number of 1's to produce the desired intensity. Obviously, no DAC is necessary in this method. Clearly, when the number of grey levels is small both the parallel and the serial methods can be used with Burst representation and then depending on speed and complexity trade-off a decision on which method to use can be made.

3.2 IS (RIS) Organization

So far it has been shown that Burst representation can be used in image pick-up and display and that the intensity information can be produced by fields where each field contributes only one unit of intensity for each image point. This suggests that IS can also be organized by fields, where each field consists of a single shift register (SR) of length N_s^2. This is shown in Figure 3.1. The video coming from the image sensor is simply shifted into the appropriate field SR one bit at a time for each sampled point, and when a field scan is complete the next field information is shifted into the following SR, and so on. Note that no encoding is required and only a demultiplexer is necessary in order to determine where the current field is being stored. However, if the parallel pick-up method is used all the SRs are shifted concurrently and the demultiplexer is eliminated. After all the image fields have been stored in the corresponding field SRs, it can easily be observed that columns of bits in all the SRs represent the (Burst) intensity of image points, e.g., the right most column in Figure 3.1 represents the intensity of point 0,0.

The display of an image stored by fields may be done in two ways, as mentioned earlier. In a serial display the field SRs are simply shifted out, one SR at a time, and then a multiplexer is necessary to steer the desired SR output into the z axis of the display unit. In a parallel display all the SRs are shifted out concurrently and the multiplexer is eliminated. However, a Burst to Analog converter is now necessary.

The operating speed of the IS SRs and the subpicture size affect the way pick-up or display are done. If the subpicture size is low, slow SRs and serial pick-up and display may be used. If large subpictures are

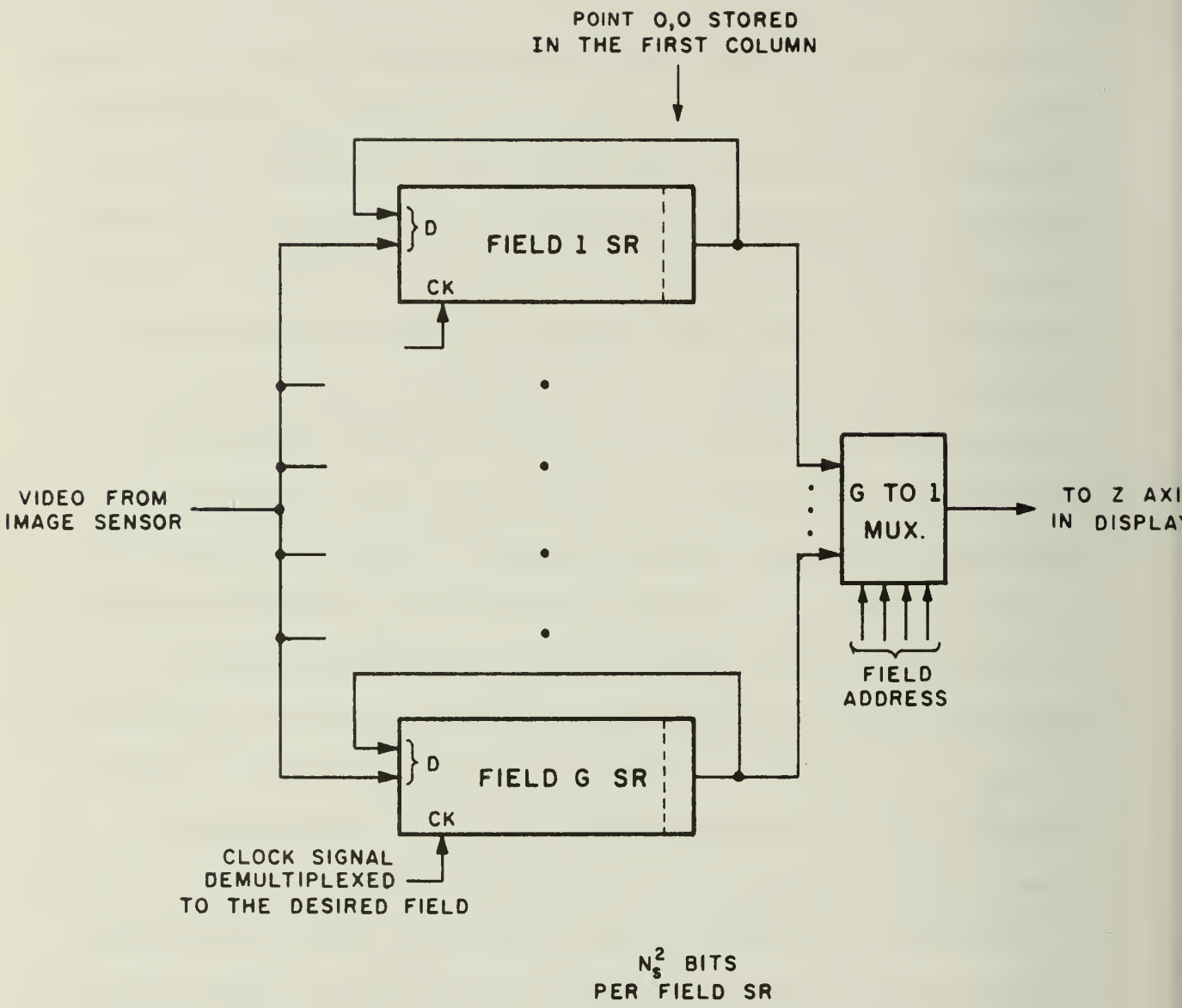


Figure 3.1 Information Store Organization

processed, it is necessary to use fast SRs since in the same amount of time necessary to display an image more points have now to be displayed. It may also be necessary to convert from serial to parallel pick-up and display in these cases.

3.3 Transformer Design and Analysis

The Transformer (T) is an implementation of equation (6), section 2.2, reiterated here with N changed to N_s:

$$Y(k,m) = \frac{1}{N_s} \sum_{1,n = 0,1,\ldots,N_s-1} \sum X(1,n) \cdot WAL(k,1) \cdot WAL(M,N) \qquad (8)$$

$$\text{for } k,m = 0,1,\ldots,N_s-1$$

and, as explained previously, the computation for each CS coefficient, Y(k,m), requires N_s^2 additions/subtractions of the individual IS X(1,n)'s. The product WAL(k,1) · WAL(m,n) (being either +1 or -1) determines whether the X(1,n) has to be added or subtracted. Consequently, T has to accept both the individual X(1,n)'s and the product of the Walsh functions and then add or subtract the X(1,n) to a temporary storage for Y(k,m), i.e., T is essentially an adder with the capability of scaling by N_s.

In Burst Processing the representation perfectly suited for such an application is the biased representation. In this representation the complement of a Burst represents the negation and so an Exclusive-OR (XOR) gate can be used to perform the addition/subtraction required. If addition is desired the Burst is passed without change and if subtraction is desired the Burst is complemented by applying a logic 1 to the other input of the XOR. For example, let G be 12, i.e., the X's are Bursts of 12 pulses. Then, 6 represents 0 and if X = -2 (which is represented by the Burst

111100000000 = 4) -X = +2; But the complement of 4 is 000011111111, which is 8 and this is exactly the representation of +2. In general, a number X is represented by (X+6) = X' and -X is represented by (-X+6) = (-X)'; But (-X+6) = 12 - (X+6) or, (-X)' = 12 - X'. The subtraction of X' from 12 is exactly equivalent to complementing X' since there are 12 - X' 0's in X' which, after being complemented, produce 12 - X' 1's which is exactly (-X)'. Therefore, the subtraction can be performed by the addition of the complement. It should be noted that for a Burst where all the pulses are available in parallel there are as many XOR gates necessary as there are Burst pulses (lines).

The main part of T is the adder and since Bursts are used, a unary adder should, obviously, be used. A basic Burst adder has been developed by Taylor [3]. It consists of a regular Full-Adder (FA) in which the carry and the sum outputs have been interchanged, hence called a Perverted Adder (PA). PA is capable of adding two Bursts and produces their sum scaled by two. (See Appendix A for a detailed description.) If more than two Bursts are to be added, a PA-Tree can be used. In such a tree every additional level gives scaling by two, and so the overall scaling factor is 2^m, where m is the number of levels in the tree. The maximum number of inputs to such a tree can be 2^m when the tree is "full", i.e., 2^m-1 PA's are used. However, in some cases it may happen that the number of inputs to the tree is smaller than 2^m and then the number of PA's required is smaller than 2^m-1. Figure 3.2 shows an example of a PA-Tree with 6 inputs and 3 levels, i.e., the scaling factor is 8. Since the number of inputs is smaller than 8 only 6 PA's are used instead of the 7 required in the full tree with 8 inputs.

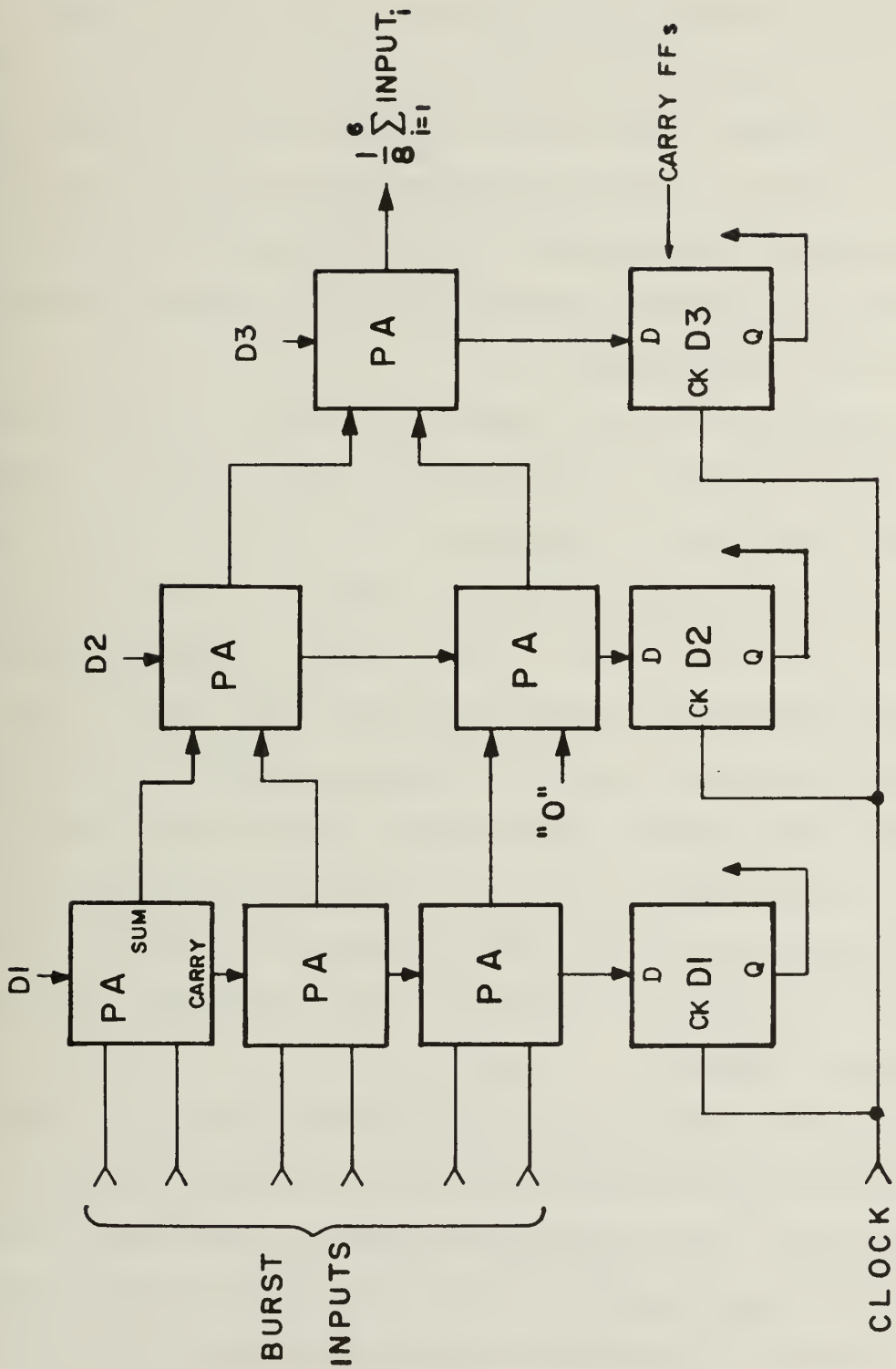


Figure 3.2 PA-Tree Diagram

In general, if the number of inputs is much smaller than 2^m it is easy to observe that the number of PA's in the last stages of the tree will decrease rapidly and only one PA will be necessary from some stage on. For example, if the tree in Figure 3.2 has still 6 inputs but a scaling factor of 16 is required, another PA level has to be added including only one PA in it. Note that the D Flip-Flops serve as carry Flip-Flops which save the intermediate remainders and, therefore, keep the computation accuracy at a maximum. (See Appendix A.)

The operation of T is as follows: The X(1,n)'s are shifted in from IS, then passed through the XORs controlled by WAL(k,1)·WAL(m,n) (which determine if the X(1,n) is complemented or not) and then passed on to the PA-Tree. The PA-Tree must have $\log_2(N_s)$ levels for a scaling by N_s. This is done for all X(1,n)'s until the computation for Y(k,m) is finished and then a new computation is started for the next Y(k,m). If X(1,n) has G grey levels (or parallel lines) G XOR gates are required. The WAL(m,n) and WAL(k,1) are generated by ROMs and their product can be computed by another XOR gate. This is possible since the Walsh functions are mapped into digital 0 and 1 and then a XOR is equivalent to a product as can be verified from a truth table with all the four combinations. (+1 is mapped into 0 and -1 into 1.)

Recall from section 2.4 that G is the number of grey levels and $N_s x N_s$ subpictures are transformed. The CS word length has been shown to be $\log_2(G \cdot N_s)$ bits. Since the biased representation is used, every addition changes the bias of the result. After N_s^2 additions the original numbers, with a bias of G/2, are changed into the CS coefficients, with a bias of $G \cdot N_s/2$, ranging from 0 to $G \cdot N_s$ (after the scaling). The inverse transform

is similar to the transform, i.e., each X'(1,n) is computed by adding or subtracting the N_s^2 Y(k,m)'s. The numbers produced by IT range from 0 to $G \cdot N_s^2$, after the scaling, with a bias of $G \cdot N_s^2/2$. However, the X'(1,n)'s must resemble the original X(1,n)'s and, therefore, must have a bias of G/2. This suggests that in IT, after the additions are performed, a correction must be made in the computed result, a correction that shifts the result from the range 0 to $G \cdot N_s^2$ to the range 0 to G. This may be done by subtraction of $(G \cdot N_s^2/2 - G/2)$ from each computed result. For example, consider the actual numbers in WALSHSTORE. Since G=12 and N_s=32, X(1,n) = A+6, where A ranges from -6 to +6. Then $Y(k,m) = B+G \cdot N_s/2 = B+192$, where B ranges from -192 to +192 , and $X'(1,n) = C+G \cdot N_s^2/2 = C+6144$, before the correction. Therefore, in order to return to C+6 a correction of -6138 is required.

Since only the Burst T has been discussed so far it is adequate now to compare it to a possible binary T, assuming that the same operations have to be performed by both systems. Figure 3.3 shows the structures of both the binary T and the Burst T. As can be observed from it a binary T requires a $\log_2(G \cdot N_s^2)$ bits FA and a $\log_2(G \cdot N_s^2)$ bit ACC register for the Y(k,m). The input is of length $\log_2(G)$ bits. Scaling by N_s is achieved by disregarding the $\log_2(N_s)$ least significant bits in ACC. The Burst T requires a $\log_2(N_s)$ levels PA-Tree with $\log_2(N_s)$ carry FF's and a $\log_2(G \cdot N_s)$ bit counter for the result. Since G, the number of inputs to the tree, is generally smaller than N_s, the tree is not full and less than N_s-1 PA's are necessary. Obviously, the number of FF's is exactly equal in both systems. As for the number of FA's (PA's) this number depends on G and on N_s.

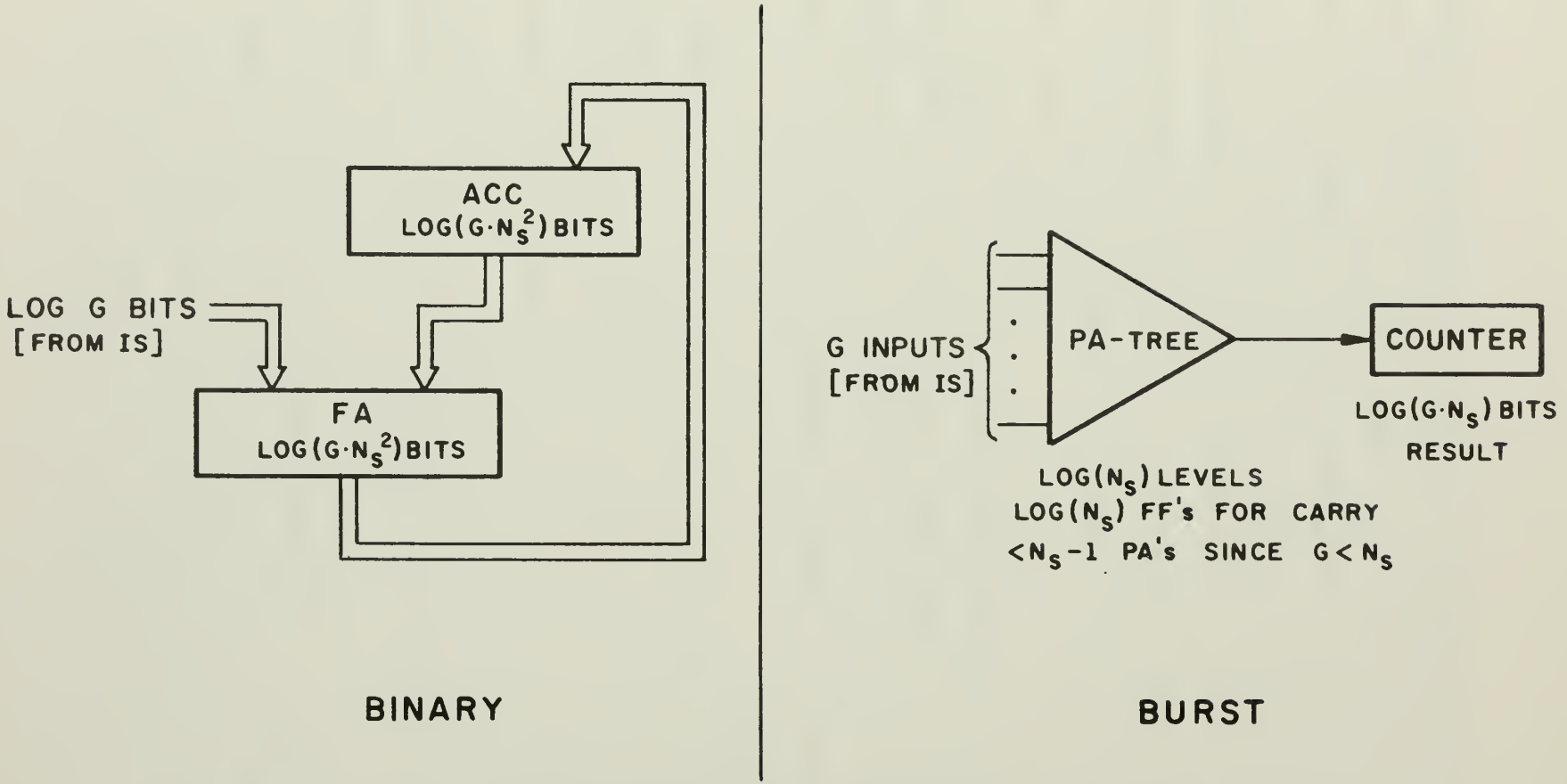


Figure 3.3 Comparison of a Burst Transformer to a Binary Transformer

In the case of WALSHSTORE the binary T requires a 14 bit FA, whereas the Burst T requires only (6+3+2+1+1) = 13 PA's. Clearly, the complexities are comparable. Additionally, if the asymptotic behaviour of both systems is analysed, it may be found that when N_s is increased by a factor of two (G remaining constant), it is necessary to add one carry FF and one PA in the Burst T and one FA and one FF in the binary T, i.e., the complexities converge. As far as speed is concerned, the performance of both systems is also comparable since both add one X(1,n) at a time.

In conclusion, in this section it has been shown how to design a Burst T with a performance comparable, in speed and complexity, to that of a binary T.

3.4 The Application to Large Pictures

In this section the applicability of the system organization, described so far in sections 2 and 3, to large pictures is determined. This covers both the general organization and the units using Burst Processing. As mentioned in section 2.4, subpictures are used in order to reduce the computation time and the amount of storage. The two different methods described there were the serial approach, where subpictures were processed sequentially, and the parallel approach in which subpictures were processed in parallel. These methods are used here, too.

First, consider the image pick-up. Since current image sensors, e.g., CCD's, are limited to clock frequencies of about 7 Mhz, the number of points being picked up by one image sensor cannot be very high. For example, for a 256x256 picture, using parallel encoding for each point assuming a

frame period of 1/30th sec., the clock signal frequency required is approximately 2 Mhz. For a 512x512 picture the clock frequency is 8 Mhz. Additionally, if serial encoding is used (as in CYCLOPS) the clock signal frequency is G times higher, i.e., 24 Mhz and 96 Mhz for the examples above. Clearly, a single image sensor cannot be used for such large pictures. There are a few solutions to this problem. If the serial encoding simplicity is desired (as in CYCLOPS), it is possible to use an array of image sensors, each covering one subpicture, and then all sensors work together and pick-up the entire picture in parallel. Since the subpicture is small it is very easy to employ current image sensors with low frequency clock signals. For example, if N_s, the subpicture size, is 16 this frequency is only 8 Khz and even if the serial encoding is used this frequency increases to 96 Khz, well within the limits of the operation of the image sensor.

For still pictures it may not be necessary to pick-up the entire picture in 1/30th sec. and the image sensors can be sampled sequentially.

IS (and also RIS) change in size when the processed pictures contain more and more subpictures. In the serial case the change amounts to using longer field SRs. In the parallel case the basic IS unit, containing G SRs of N_s^2 bits, is used as many times as there are subpictures. Clearly, the organization of IS does not change. Note that there is a strong tie between the choice of an image pick-up method and an IS subpicture storage, e.g., when an array of image sensors work in parallel IS must also be organized as an array of stores, each of the size of a subpicture.

As far as T is concerned there is no change in the basic unit described in section 3.3. This is so because T has been designed for subpictures only and its structure does not change with the number of subpictures.

In the serial case T transforms subpictures sequentially and in the parallel case there as many T's as there are subpictures and all work in parallel.

The argument applied for the growth of IS holds also for CS, i.e., in the serial case the transformed subpictures are stored sequentially in longer SRs, and in the parallel case CS is decomposed into small subsystems, each resembling the original CS for one subpicture.

As for IT, since it inverse transforms subpictures only, its structure does not change with the number of subpictures, exactly like T.

In conclusion, it has been shown that WALSHSTORE system organization applies to small, as well as large, pictures and that Burst Processing is favorable in such a system for reasons of simplicity of encoding and processing. Additionally, the system is modular and can be used with different picture sizes with small changes in the system.

4. MACHINE DESCRIPTION AND RESULTS

This section is composed of two parts. The first part provides a detailed description of the various units in WALSHSTORE, at the block diagram level, as constructed by IEL. These are the Camera and Display Control, the Information Store, the Transformer, the Convolution Store, the Inverse Transformer, the Retransformed Information Store and the Control Unit. The second part compares some software simulation results to the actual results from WALSHSTORE as a demonstration of the fail-soft property.

The parameters used in the actual machine are: $N_s = 32$, $G = 12$. For budgetary reasons only one subpicture is used, i.e., $N = 32$. The image sensor has 32x32 points and the clock signal required by it is of a frequency of 0.5 Mhz. The same clock signal is used for all other purposes, although much faster clock signals can be used since the design is based on the TTL logic family.

WALSHSTORE has five modes of operation, some controlled from the front panel. These are the IS Display mode, the Input mode, the Transform mode, the Inverse Transform mode and the RIS Display mode. IS Display and RIS Display modes are continuous and the machine enters into them at power-on. Input mode may be selected from the front panel and then the picture being picked up is displayed on the IS display unit continuously. The Transform and Inverse Transform modes are selected from the front panel and both are momentary, i.e., only one transform is performed at each time.

4.1 Camera and Display Control

Figure 4.1 shows the Camera and Display Control block diagram. The main part consists of the X, Y address counters and the field address counter. These counters are clocked by the system clock signal and they keep track of the point being picked up or displayed and on the field being accessed. Both X and Y address counters are 5 bits long since a 32x32 picture is either picked up or displayed. They feed two 5 bit DAC's which supply the deflection signals to the display units (both IS and RIS). The picture is scanned in horizontal lines, and after each line is completed the scan is moved down by one line and the horizontal scan begins again. After the two 5 bit counters complete one full cycle (1024 clock periods), the field counter is advanced by one and the scan starts all over. The 4 bit field address is supplied to other units in WALSHSTORE and it is called DISPADDR. When the field address counter completes its cycle (16 fields) and returns to field 0, the opeartion of all counters is stopped for 4 μsec. This is done by switching on the Frame Sync monostable, which in turn asynchronously clears the three counters. The FSYNC signal, produced by this monostable is used as a sync signal for mode changes, so that machine modes may change only after a complete picture scan has been performed.

The camera clock signal (CAMCK) is also controlled by FSYNC. CAMCK is essentially the system clock signal and it is used to advance the address counters in the camera, counters which access the point being sampled. (These counters run in exact synchronism with X, Y address counters here.) The camera also requires a 4 μsec. reset signal every 16 fields and this signal is produced by superimposing FSYNC on the system clock signal by means of AND gate 1. Note that during FSYNC the field

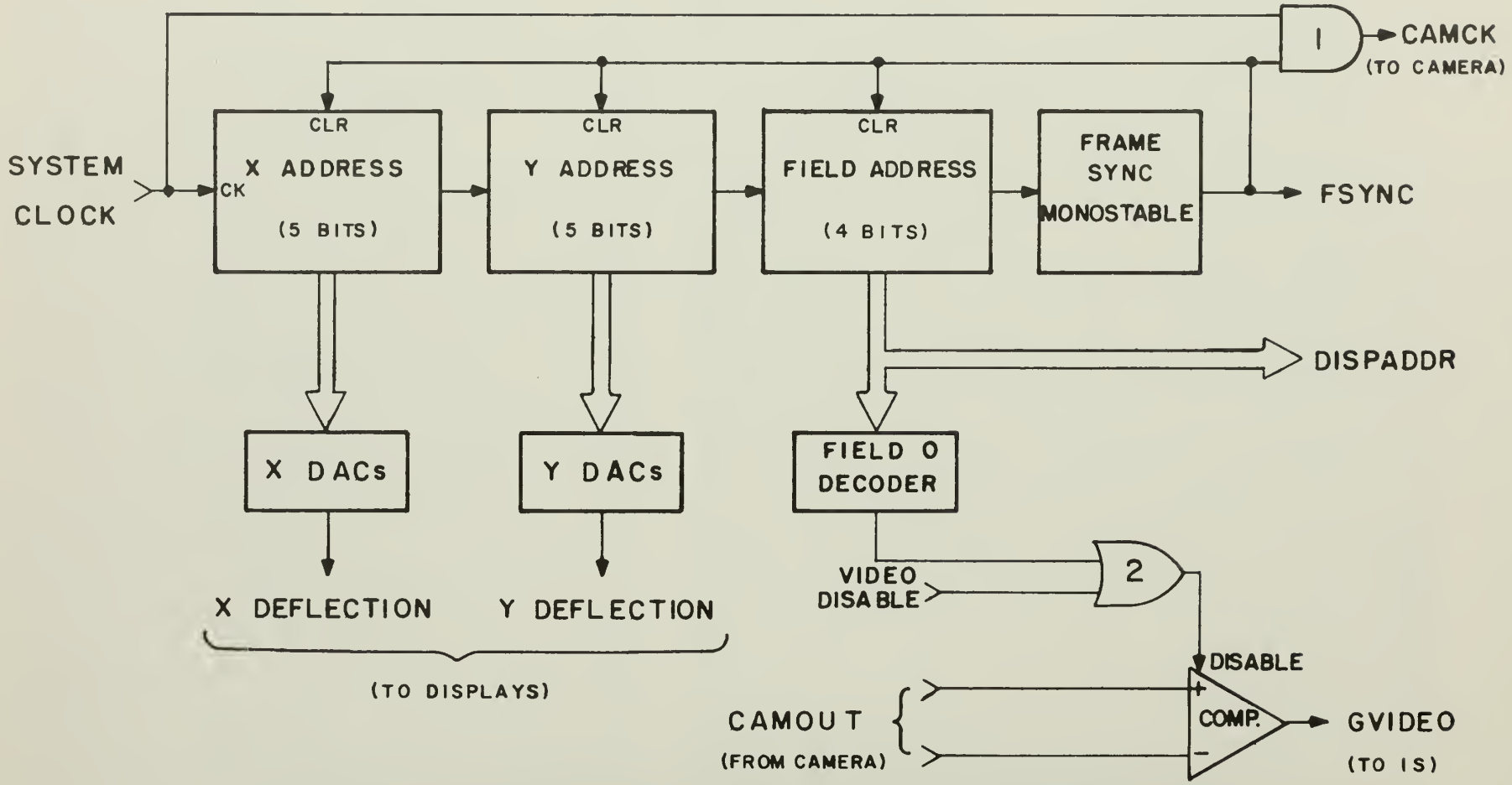


Figure 4.1 Camera and Display Control Block Diagram

address is kept at 0 and that no meaningful video is produced by the camera.

The video signal arrives from the camera as a differential pair (CAMOUT) so a comparator is used to convert it to TTL levels. The comparator is disabled during VIDEO DISABLE and during field 0. VIDEO DISABLE disables the comparator during the switching of the camera between adjacent points and so reduces the noise passed on to IS on the GVIDEO line. Since during field 0 the camera is being reset, a field 0 decoder is used to disable the comparator during this period and it overrides VIDEO DISABLE by means of OR gate 2.

4.2 Information Store

The IS block diagram is shown in Figure 4.2. The heart of IS is the 12 1K-bit field SRs which are used to store the information supplied by the camera. This information may then be displayed or used in the transform. The IS can operate in three modes: Input, IS Display and Transform.

In the Input mode the fields being received from the camera are stored in the field SRs, one field at a time, and are also displayed ON-LINE on the IS display unit. The SRs' clock signal ISCK is demultiplexed into the desired field SR by means of a 1 to 12 ISCK demultiplexer controlled by DISPADDR. ISCK passes without change through the TCK Override Circuit to the correct SR. The INPUT signal has two functions. First, it selects GVIDEO, coming from the Camera and Display Control, as the data input to the field SRs, instead of their own recirculating outputs. Second, it selects GVIDEO as the video to be displayed on the IS display. This is done by the 2 to 1 Tri-State wired-AND Z-axis multiplexer. This data passes through a NOR gate to the actual Z-axis in the IS display.

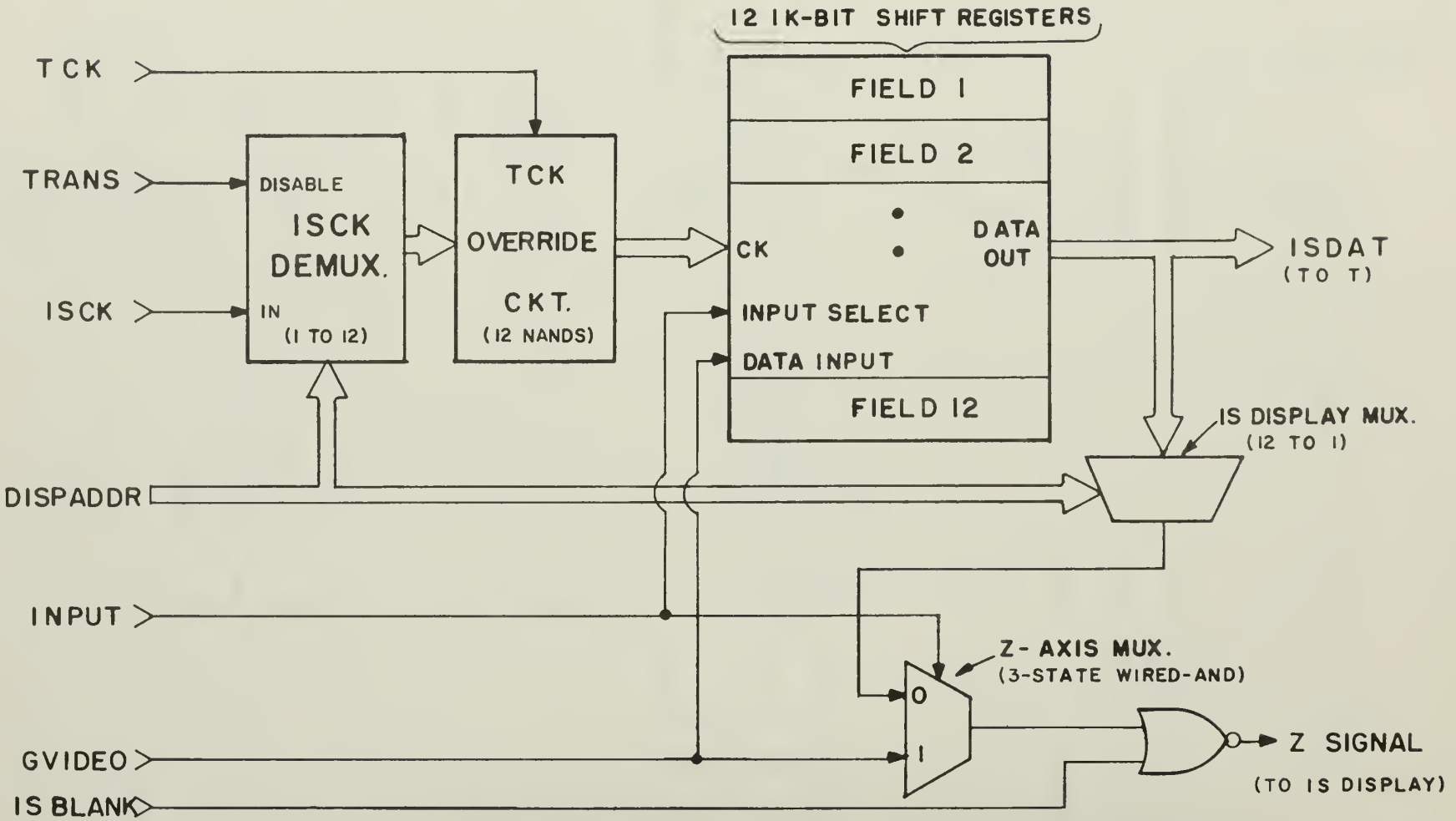


Figure 4.2 Information Store Block Diagram

The IS BLANK signal makes sure that a point is displayed only after all the switching transients of the deflection DAC's have settled. Note that 16 fields of information are displayed exactly as received from the camera but only 12 are stored in IS since G is 12.

In the IS Display mode the information stored in IS is displayed on the IS display unit. ISCK operation is exactly the same as that in Input mode. Since INPUT is not active now, the data inputs to the field SRs are their own recirculating outputs and, therefore, their contents are not lost in this mode. Additionally, the Z-axis multiplexer selects now the Z-axis information coming from the 12 to 1 IS display multiplexer which in turn selects one of the field SR outputs (ISDAT) as dictated by DISPADDR. Note that although DISPADDR cycles through 16 field addresses, only 12 are meaningful, i.e., 1 to 12, and the rest are displayed as 0's.

In the Transform mode the operation of IS is different from the operation is the Input and the IS Display modes. Since T requires full columns of Bursts, all the IS SRs have to be recirculated <u>together</u> and then ISDAT do not have any meaning for display purposes. Consequently, IS BLANK is kept active during the entire transform period and blanks the IS display. Additionally, the ISCK demultiplexer is disabled during Transform mode and the Transform clock signal TCK is passed to all the SRs through the TCK Override Circuit. ISDAT is transmitted, one point at a time, to T on 12 parallel lines.

After the transform is completed the machine returns automatically to IS Display mode. However, there is no synchronization between TCK and ISCK, i.e., ISCK is in the middle of a cycle when TCK completes its cycle and the

contents of the SRs are at the original positions. Consequently, it is necessary to wait until FSYNC and then actually return to IS Display mode.

4.3 Transformer

The Transformer block diagram is shown in Figure 4.3. T's main part consists of the PA-Tree and the Y(k,m) counter, as discussed in section 3.3. The Walsh ROMs receive their addresses (k,1,m,n) from the Control Unit and produce two 1-bit Walsh functions, i.e., WAL(k,1) and WAL(m,n). These are XORed to produce the TAD/SB signal which determines if the current X(1,n) should be added or subtracted. At the same time ISDAT (the Burst representation of X(1,n)) are received by the 12 line-receivers and passed to the Transfer/Complement Network consisting of 12 XORs. Depending on TAD/SB, this network either passes X(1,n) as it is (for addition) or complements it (for subtraction). The Burst is then fed into the PA-Tree which is a 5 level, 12 input PA-Tree which scales by 32. The output from the tree is clocked into the Y(k,m) counter by means of TCK, which also clocks the carry FF's in the tree itself. After the computation period for the Y(k,m) ends the Y(k,m) is loaded into a 9 bit SR and then shifted out serially into the corresponding CS quadrant via the TDAT line. This is done by the Transform Output clock signal TOUTCK. TOUTCK also clears the carry FF's in the PA-Tree as a preparation for the next Y(k,m) computation.

T, obviously, operates during the Transform mode. In this mode one transform is performed for each command from the front panel. As mentioned earlier, the IS SRs are recirculated in order to supply all 1024 X(1,n)'s to T. This is done by TCK too. TCK has a period of 1033 system clock periods. During the first 1024 clock periods, TCK shifts one X(1,n) at a

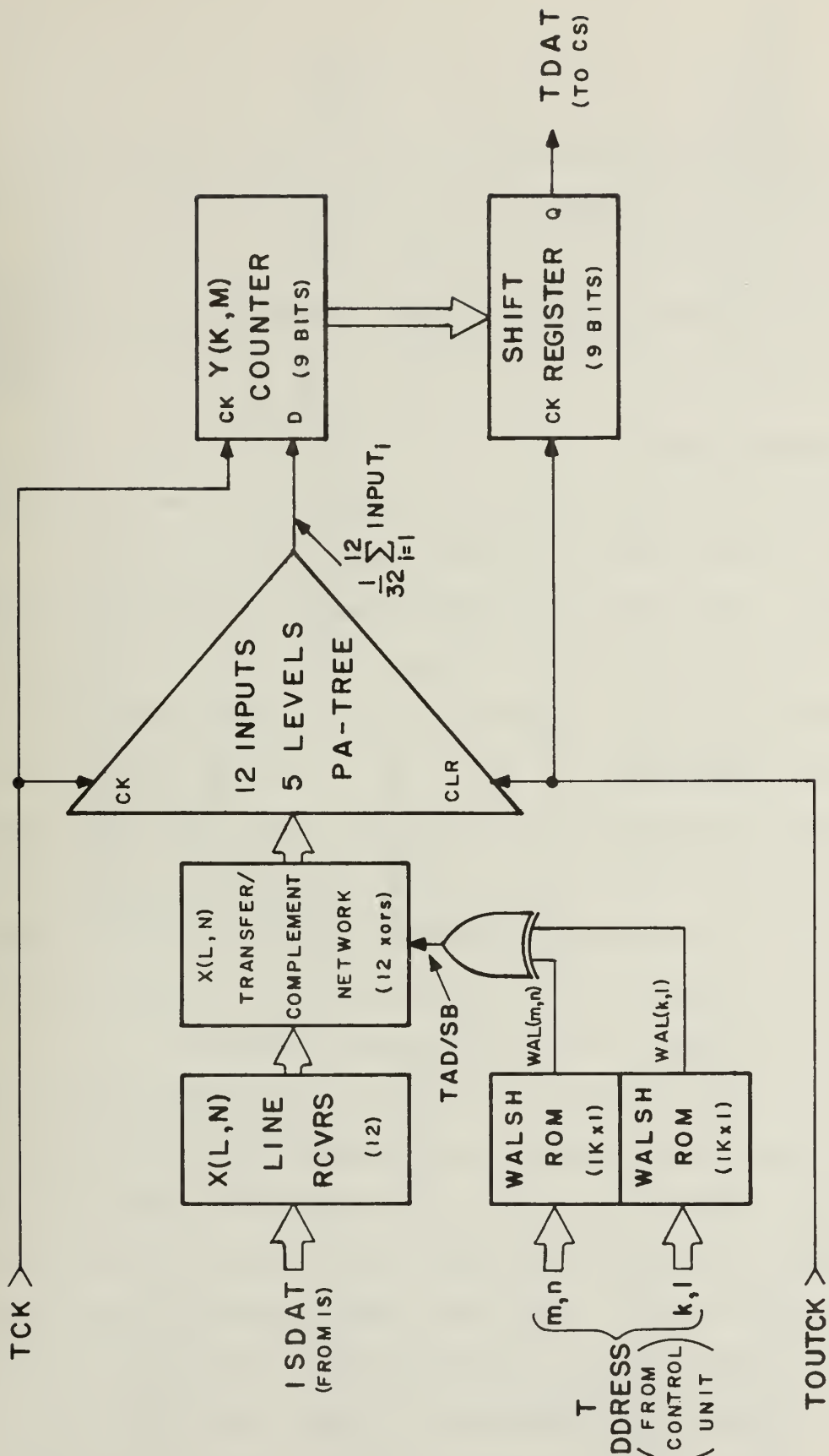


Figure 4.3 Transformer Block Diagram

time into T inputs and also clocks T itself. After this period TCK is deactivated for 9 clock periods during which TOUTCKeffectuates the Y(k,m) into CS. This operation is repeated 1024 times for all Y(k,m)'s and afterwards the Transform mode is exited. Note that TCK and TOUTCK originate at the Control Unit.

4.4 Convolution Store

CS consists of five quadrants (four quadrants and one duplicate of the first quadrant), and each quadrant stores 8 rows of transform coefficients. The quadrants are interchangeable because they are exactly identical and the only difference between them is their physical location in WALSHSTORE. Therefore, only the operation of the first quadrant is described in this section.

Figure 4.4 shows the CS quadrant block diagram. The central part consists of the 8 288-bit row SRs. These are recirculating SRs except when data is to be entered into them from the TDAT line. The SR length is a result of the need to store 32 9-bit coefficients in each row SR. CS operates in two modes: Transform and Inverse Transform.

In the Transform mode CS receives the Y(k,m)'s coming from T on TDAT. Since the TRANS line is active, the SRs change from recirculating mode to input mode and TDAT is entered into the desired row SR. Eight CSCK lines, CSCK0-CSCK7, are provided by the Control Unit in order to effectuate TDAT into one of the row SRs. These clock signals are demultiplexed into the correct line by means of a 1 to 32 demultiplexer located in the Control Unit and addressed by the coefficient's k index. (Note that 24 more lines exist for the other 3 quadrants.) The CSCKs are passed through the CSCKIT

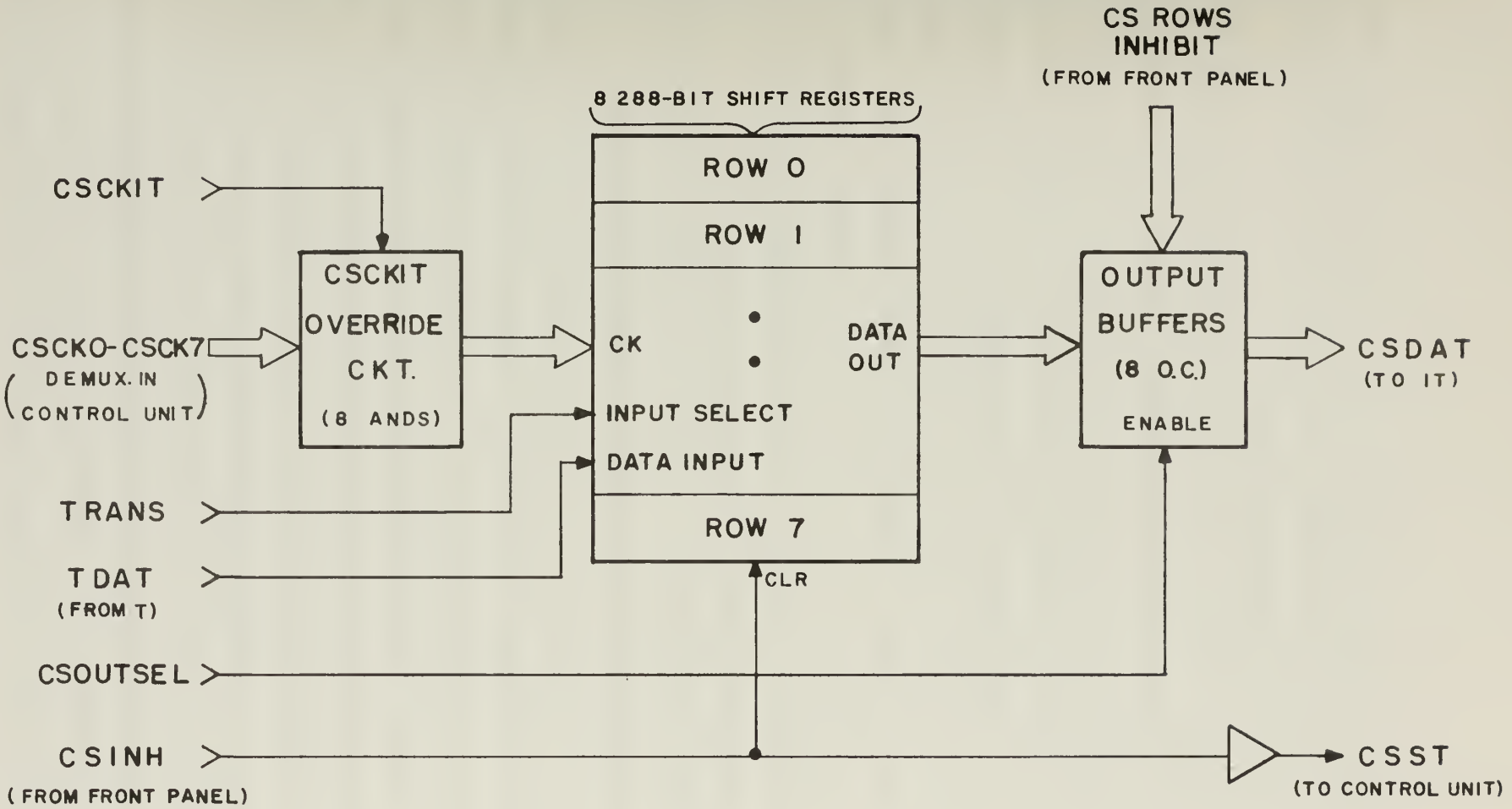


Figure 4.4 Convolution Store Quadrant Block Diagram

Override Circuit without change in this mode. Note that the CSCK signal is essentially TOUTCK described earlier. Note, also, that CSCK0-CSCK7 are provided to both first quadrant duplicates.

In the Inverse Transform mode all the row SRs outputs are used in parallel. Therefore, all SRs are recirculated together. This is done by means of the CS Inverse Transform clock signal, CSCKIT, which overrides CSCK0-CSCK7 by means of the CSCKIT Override Circuit. The SRs' data outputs are buffered for noise immunity and then transmitted to IT on the CSDAT lines. (Every quadrant has 8 such lines.) The output buffers are enabled with CSOUTSEL, which is active only during the Inverse Transform mode. Note that CSDAT are designed with open collector gates in order to be able to multiplex the contents of the first quadrant duplicates on the same lines (wired-OR) without the need of an actual multiplexer.

The signals not described so far are: CSINH, CSST and CS ROWS INHIBIT. They are concerned with the simulation of loss of whole quadrants, transmission of status information and the simulation of loss of individual rows in the first quadrant, respectively. Quadrant loss may be simulated in two ways, i.e., physical removal of the quadrant card from the machine or use of a switch located on the front panel to switch the quadrant off. The CSINH signal is used to notify the quadrant that loss simulation is to be made. This signal simply clears the row SRs in the quadrant and their contents are lost. The CSST signal notifies the Control Unit about the quadrant's status. In the first quadrant case this signal is used to determine which quadrant is to be enabled during Inverse Transform mode by means of the CSOUTSEL lines. In the other quadrants CSST is only used for status display purposes. Note that, regardless of the way a loss is achieved,

the CSST always carries the correct status information. This is achieved by using a logic 1 as the loss status and then when a card is removed the TTL input on the other side of the line interpretes it also as a logic 1. The CS ROWS INHIBIT lines are provided for experimental purposes and they can be used to destroy individual rows in the first quadrant. These lines are controlled by 8 switches located on the front panel and they simply block the corresponding output buffer during Inverse Transform mode and so simulate a loss of a row. Note that this loss is not permanent since the row information is still stored in the row SR.

4.5 Inverse Transformer

IT performs the same opeartions that T does, i.e., addition or subtraction of individual Y(k,m)'s to produce each X'(1,n). The Y(k,m)'s are stored in 9-bit binary words but they use the biased representation so that subtraction may be performed by addition of the complement. Consequently, XOR gates can be used, like in T, to do the addition or subtraction and they are controlled by WAL(k,1) and WAL(m,n), again. Recall from section 3.3 that a correction is necessary after the inverse transform is performed in order to map the result into the range -6 to +6. This correction has been shown to be -6138 for the parameters used here. However, since the Y(k,m)'s range from 0 to 384, i.e., 9-bit binary, and the complement is not relative to 384 but to 511 (all 1's), the range of the result changes and a different correction constant must be used. In order to compute this constant it may be observed that, except for X'(16,16), all X'(1,n)'s require exactly 512 additions and 512 subtractions. (This results from the Walsh functions property of having equal numbers of -1's

and +1's.) Additionally, the Y(k,m)'s have a bias of 192 and after being complemented relative to 511 this bias changes to 319 since the numbers are in the range of 127 to 511 now. Consequently, X'(1,n)'s bias before the correction and after scaling by 32 is: $512 \cdot 192/32 + 512 \cdot 319/32 = 8176$ and, therefore, a correction of -8170 is necessary in order to return to the range -6 to +6. Note that in X'(16,16)'s case only additions are performed and, therefore, the correction constant remains -6138.

As mentioned earlier the CS organization is such that a complete column of coefficients is accessible and can be processed in parallel. This is used in IT to speed up the computation. The equation to be implemented is:

$$X'(1,n) = \frac{1}{32} \sum_{k,m = 0,1,\ldots,31} \sum Y(k,m) \cdot WAL(k,1) \cdot WAL(m,n)$$

where k represents the rows and m the columns in Y matrix. Since m is fixed for the column it is observed that WAL(m,n) is also fixed for all Y(k,m)'s in the same column and then each Y(k,m) has to be multiplied by the corresponding WAL(k,1). Consequently, Walsh functions WAL(m,n) and WAL(k,1) (for all k's) have to be supplied to IT and then parallel addition/subtraction can be performed on columns. As described earlier, multiplication can be performed by XORs and then two XORs are necessary for each Y(k,m) in the processed column. One XOR performs the multiplication of WAL(k,1) and WAL(m,n) and its output is used to control the Y(k,m). Obviously, the WAL(k,1) ROM required is 32x32 bits and then 64 XORs are also necessary. However, since WAL(m,n) is common to all the 32 XORs and its function is only to control the complementing of each WAL(k,1), this function can be incorporated into the WAL(k,1) ROM by simply increasing its size to 64x32 bits and using

WAL(m,n) as the most significant address bit in addition to 1. The lower half of the ROM stores the 32 WAL(k,1)'s and the upper half stores their complements and then only 32 XORs are necessary to multiply each Y(k,m) with the WAL(k,1) · WAL(m,n) provided by the ROM.

Figure 4.5 shows the IT block diagram. The Walsh ROMs receive their addresses (1,m,n) from the Control Unit and the WAL(m,n) ROM output is used to address the WAL(k,1) ROM. The latter ROM's output lines, ITAD/SBO-31, are fed into the Transfer/Complement Network and they determine whether the 32 CSDAT are added or subtracted. The addition is done bit-serially on each column of coefficients. The sum of all the 32 equally weighted bits is converted into a 6-bit word, called TREE. This word is added to the partial result register, PR, which is then shifted right by one position so the next TREE word, representing the next significant bit sum, can be added correctly. After 9 such additions and shifts, during which all the first column coefficients have been added, multiplexer 1 is programmed to select the accumulator ACC instead of TREE and PR is added to ACC. Then PR is cleared and the operation repeats 31 more times. At this time, called ITOUT, the computation of the current X'(1,n) has ended and ACC contains the result before the correction. The next clock period ACC is selected in multiplexer 1, and C1 or C2 is selected in multiplexer 2, and the correction is performed by the addition of C1 or C2 to ACC. C1 = -8170 for all X'(1,n)'s except for X'(16,16) where C2 = -6138 is used. In the second clock period during ITOUT the corrected result is truncated and sent to RIS on the ITDAT lines. After that ACC is cleared and the next X'(1,n) computation is started. (Note that the CS SR contents are exactly in their original

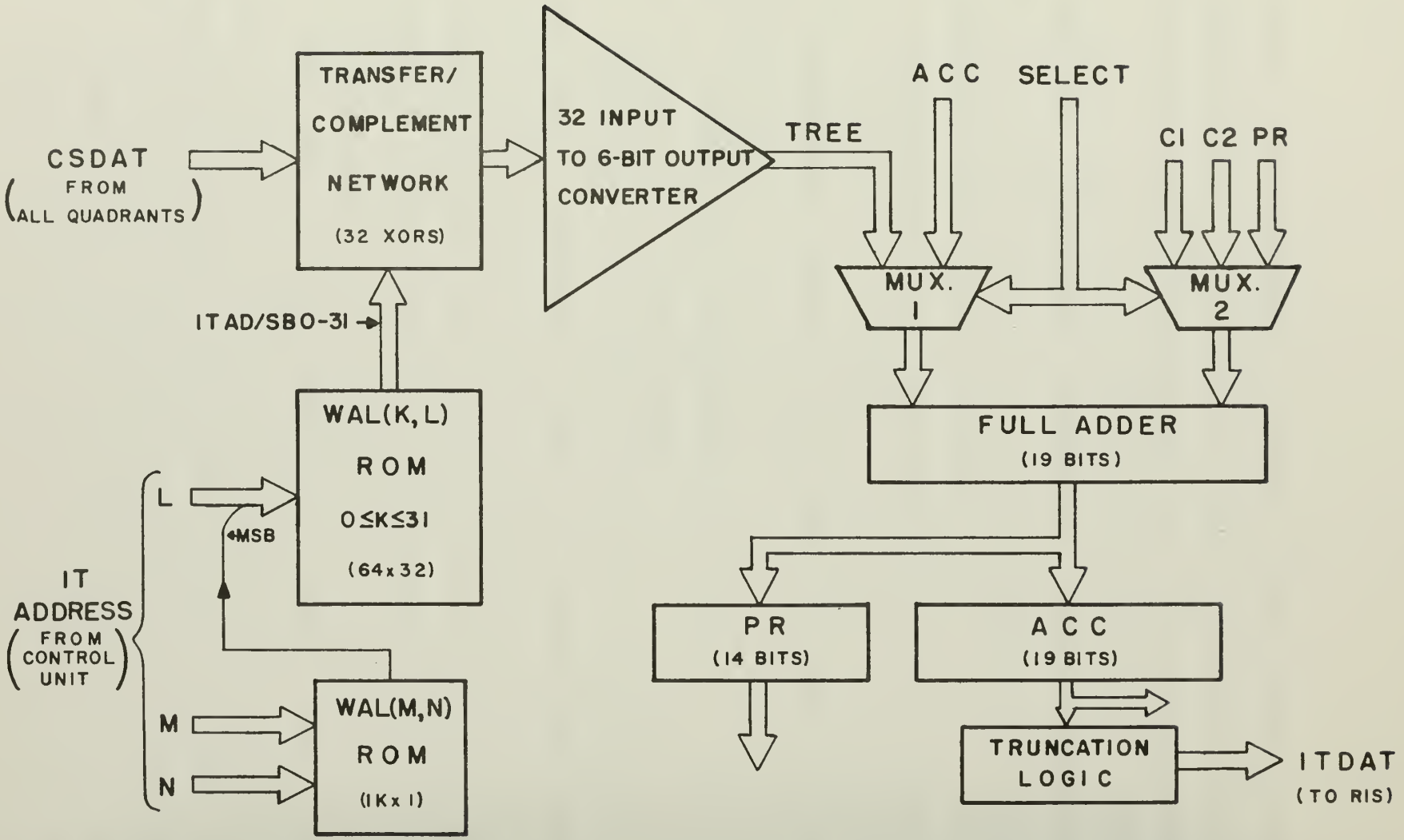


Figure 4.5 Inverse Transformer Block Diagram

position now since they are recirculated.) This cycle is repeated 1024 times for all X'(1,n)'s and then Inverse Transform mode is exited. The machine, however, returns to RIS Display mode only at the next occurrence of FSYNC.

4.6 Retransformed Information Store

RIS's operation and structure are similar to those of IS. Figure 4.6 shows the RIS block diagram. The 12 field SRs store the information received from IT and then may be displayed on the RIS display. RIS operates in two modes, namely, Inverse Transform and RIS Display. In the Inverse Transform mode ITDAT, received from IT, are converted from binary to Burst and then shifted into all the field SRs in parallel, by means of the RISCK. This conversion is done so that both IS and RIS have the same data format and the same control signals can be used for display purposes. During this mode ITRANS changes the normally recirculating SRs into input mode. In RIS Display mode the SRs are recirculated and, similar to IS Display mode, one field is displayed at a time. This is done by selecting one field SR output by means of the RIS display multiplexer addressed by DISPADDR. The actual Z-axis signal passes through a NOR gate whose other input is the RIS BLANK signal. This signal's function is similar to IS BLANK's function, i.e., to blank the RIS display during the Inverse Transform mode when the field SRs' data outputs are not meaningful, and to blank the display during the deflection DACs' transients in the RIS Display mode. Note that RISCK in RIS Display mode is exactly identical to ISCK in IS Display mode.

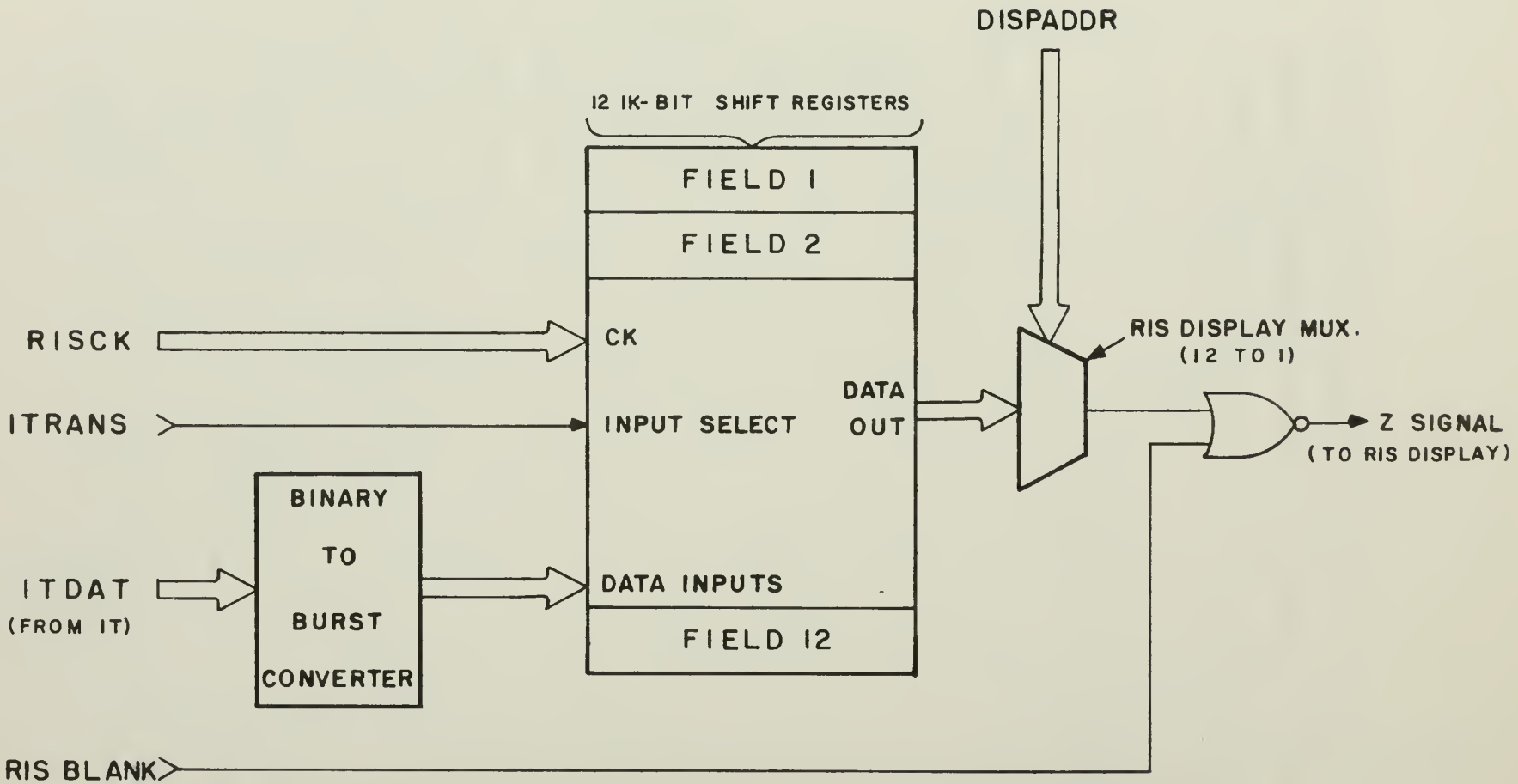


Figure 4.6 Retransformed Information Store Block Diagram

4.7 Control Unit

The Control Unit provides the necessary control signals to all units described earlier and it includes the mode selector, the clock signal generator, the T/IT controller and the failure and status controller, as shown in Figure 4.7.

The mode selector is activated by push-buttons located on the front panel. Mode changes, as already mentioned, are performed only at FSYNC so that all clock signals are always in synchronism. Mode outputs are displayed on the front panel and are also used in the general control logic which determines what control signals to produce during each mode.

The clock signal generator includes a free running multivibrator which produces a 2 Mhz symmetric square waveform, and a divide by four network which divides this 2 Mhz clock signal into four nonoverlapping clock phases. Each clock phase has a period of 2 μsec. and is active for 0.5 μsec. Note that the clock signal generator provides the clock signal for the Camera and Display Control.

The T/IT controller operates during Transform or Inverse Transform modes, respectively. This circuit is essentially a large counter, with some additional logic gates, which keeps track of the point accessed and the coefficient being computed. During the Transform mode there are 1024 clock periods necessary in order to compute one transform coefficient and there are 1024 such coefficients. Consequently, a 20-bit long counter is required, which is composed of four 5-bit counters and is enabled by TRANS, as shown in Figure 4.7. During the Inverse Transform mode there are 320 clock periods necessary in order to compute one inverse transformed point, (10 clock periods for one columns and 32 columns) and there are 1024 such points.

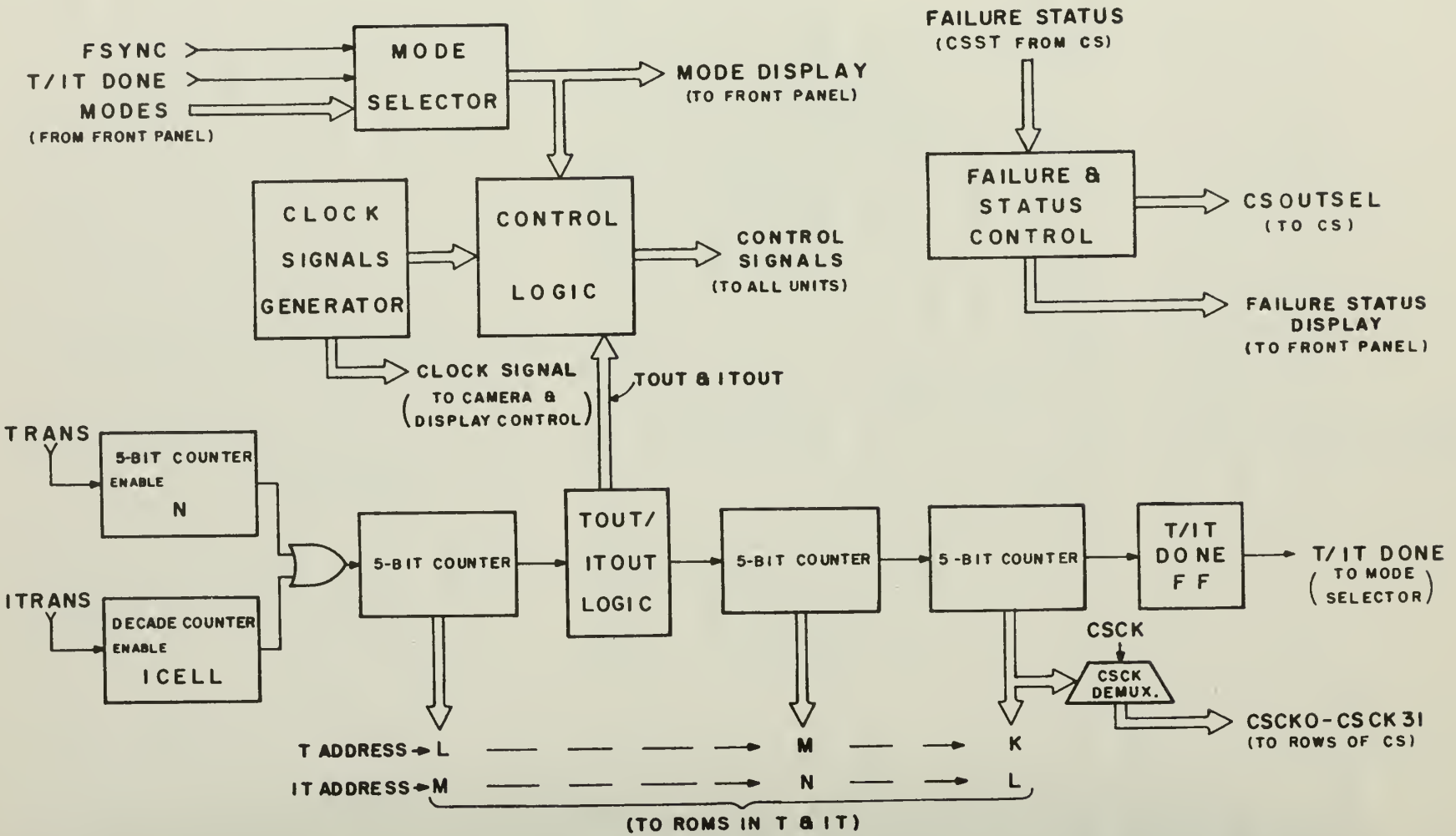


Figure 4.7 Control Unit Block Diagram

Consequently, a 19-bit long counter is required which can be implemented using the same counter used in the Transform mode. The only change is in the first counter, which is now a decade counter called ICELL. This 19-bit counter is enabled by ITRANS. Note that the same counter may represent a different index during different modes, as illustrated in Figure 4.7.

After each transform or inverse transform of a point is completed, the machine enters into TOUT or ITOUT states, respectively, and performs an output operation. This is done by means of the TOUT/ITOUT logic which is activated after each complete cycle of the first two counters. After the whole transform or inverse transform is completed the T/IT DONE FF is set and then this signal is used to exit the corresponding mode.

The control signals to all units during all modes are generated by the control logic. The exception is the CSCK signals. These are generated by demultiplexing CSCK, during Transform mode, with a 1 to 32 demultiplexer addressed by the coefficient's index, i.e., the row index, and so each CS row SR receives its own clock signal when the corresponding row is being transformed.

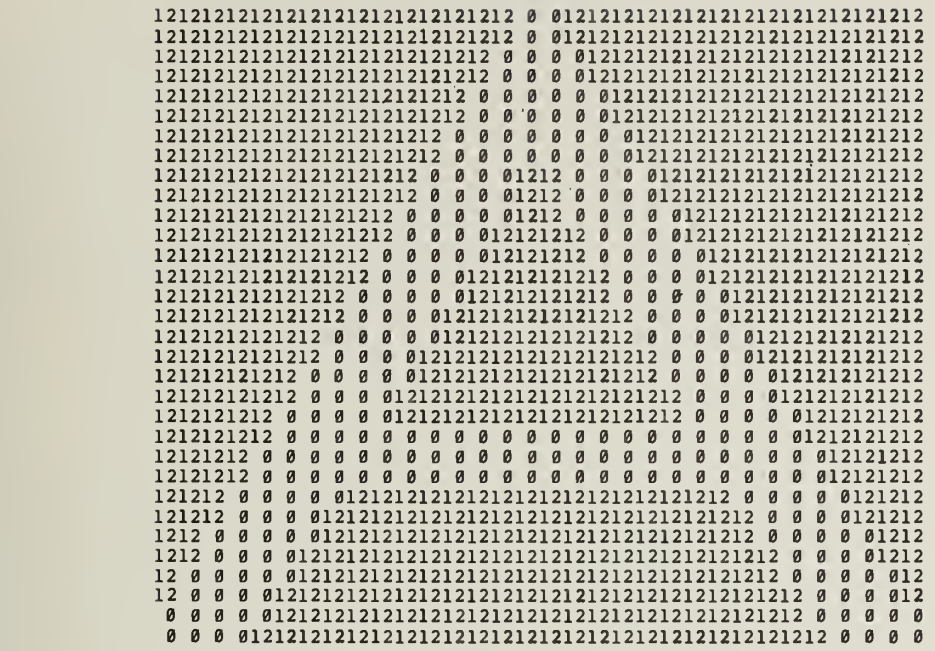
The failure and status control receives the status information from all CS quadrants (CSST), determines which one of the duplicates is operational and then sends the CSOUTSEL enabling signals to all quadrants. When both duplicates are operational, CSOUTSEL for CS11 is arbitrarily chosen. The failure and status control also supplies the status information to the status display located on the front panel.

4.8 Results

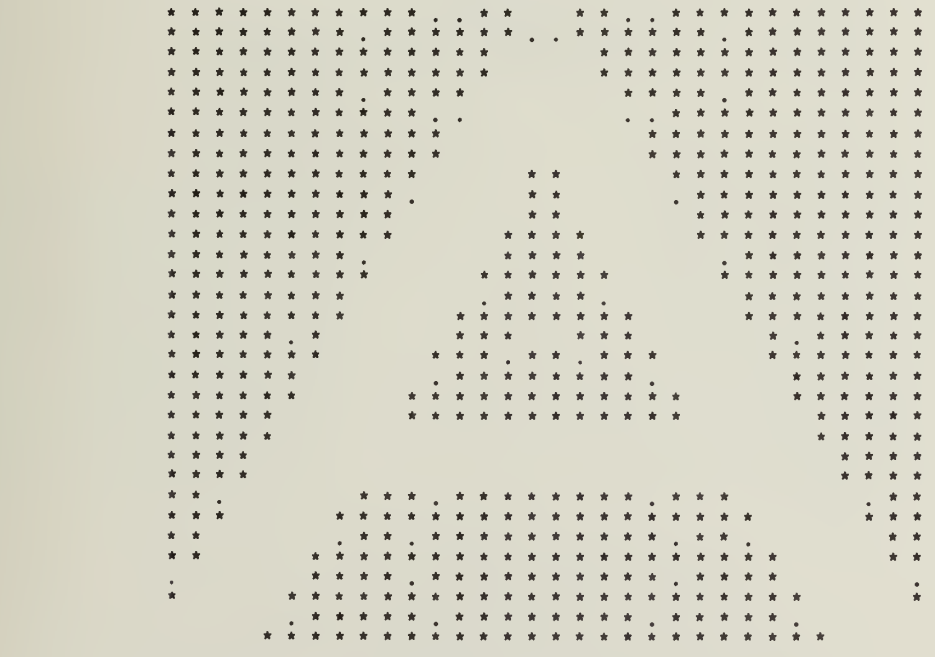
This section provides a comparison between WALSHSTORE's software simulations and the actual operation of the machine. The picture illustrated is of the letter A, and a simulation of the behaviour of the retransformed picture with no loss and with loss of single quadrants has been performed, as shown in Figure 4.8. Figure 4.9 shows photographs of the actual display units with the same letter A being processed. In addition to the actual picture output the absolute mean error between the original and the inverse transformed pictures has been measured both in the simulation and in the actual machine.

As may be observed from Figures 4.8 and 4.9 the loss of the first quadrant is intolerable both in terms of the resulting picture and the amount of error. Also, when no loss occurs, both Figures show that the errors due to scaling and truncation are negligible and that the absolute mean error is less than one unit of intensity. It should be noted that other patterns were also simulated and tested and they showed similar behaviour.

INFORMATION STORE, IS(L,N)



RETRANSFORMED INFORMATION STORE WITHOUT LOSS, RIS(L,N)
BLANK=0 TO 4, .=5 TO 9, *=ABOVE 9

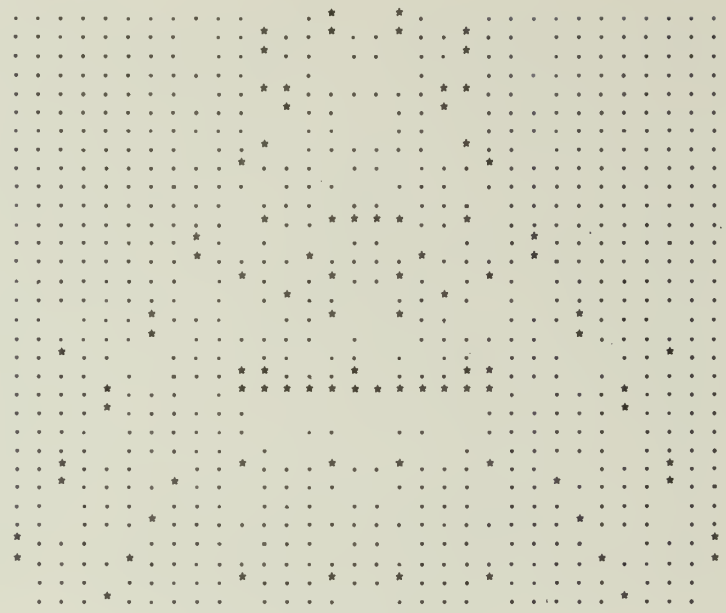


THE MEAN ERROR DEVIATION FOR THIS CASE IS:

0.500

Figure 4.8 (a) Software Simulations for the Pattern A

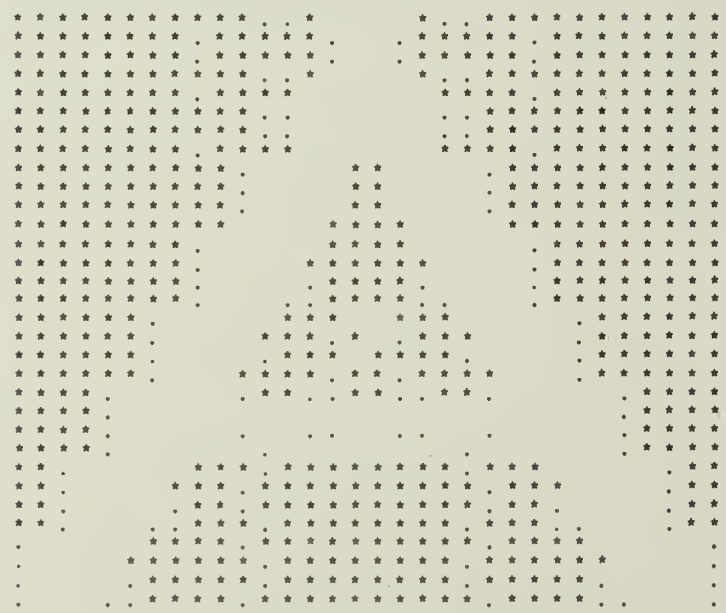
QUADRANT 1 DELETED FROM CONVOLUTION STORE
RETRANSFORMED INFORMATION STORE, RIS(L,N)
BLANK=0 TO 4, .=5 TO 9, *= ABOVE 9



THE MEAN ERROR DEVIATION FOR THIS CASE IS:

5.271

QUADRANT 2 DELETED FROM CONVOLUTION STORE
RETRANSFORMED INFORMATION STORE, RIS(L,N)
BLANK=0 TO 4, .=5 TO 9, *= ABOVE 9

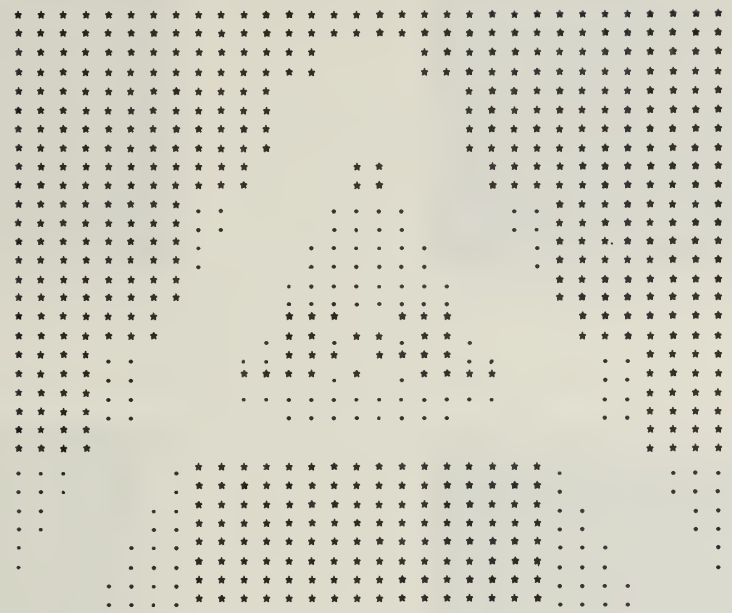


THE MEAN ERROR DEVIATION FOR THIS CASE IS:

1.102

Figure 4.8 (b)

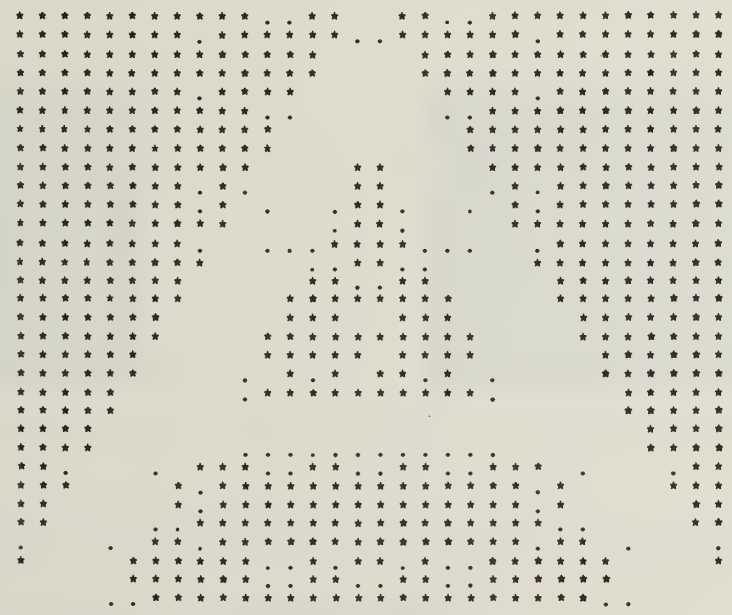
QUADRANT 3 DELETED FROM CONVOLUTION STORE
RETRANSFORMED INFORMATION STORE, RIS(L,N)
BLANK=0 TO 4, .=5 TO 9, *= ABOVE 9



THE MEAN ERROR DEVIATION FOR THIS CASE IS:

1.471

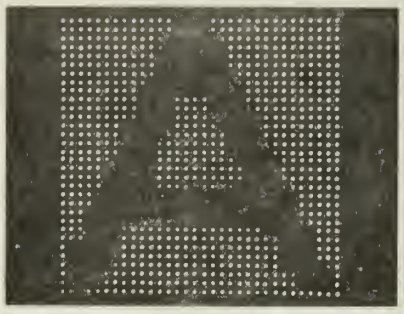
QUADRANT 4 DELETED FROM CONVOLUTION STORE
RETRANSFORMED INFORMATION STORE, RIS(L,N)
BLANK=0 TO 4, .=5 TO 9, *= ABOVE 9



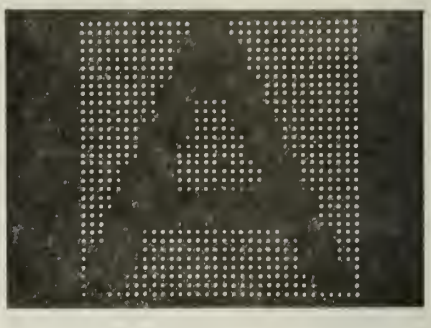
THE MEAN ERROR DEVIATION FOR THIS CASE IS:

0.942

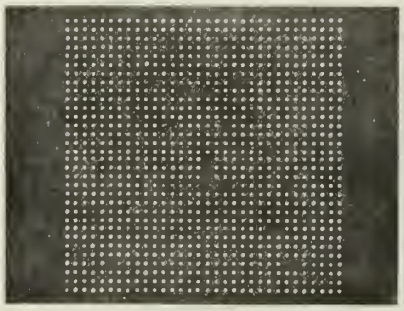
Figure 4.8 (c)



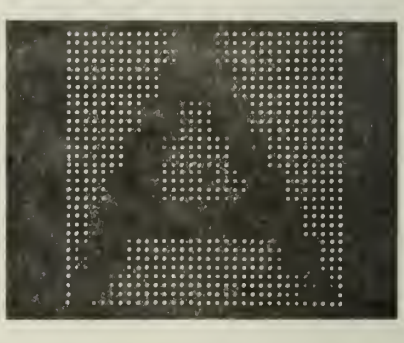
(a)



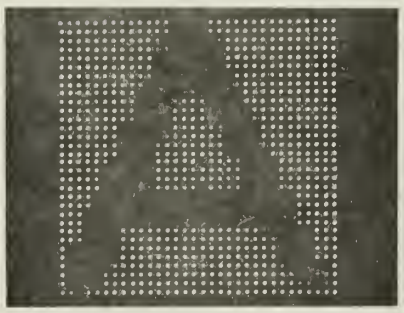
(b) Mean Error = 0.610



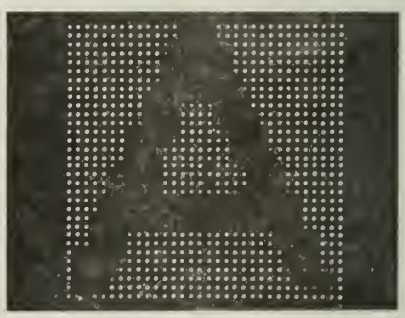
(c) Mean Error = 5.801



(d) Mean Error = 1.084



(e) Mean Error = 0.850



(f) Mean Error = 1.218

Figure 4.9 Results from WALSHSTORE for the Pattern A

(a) Original Picture (b) No Loss

(c) Loss of Quadrant 1 (d) Loss of Quadrant 2

(e) Loss of Quadrant 3 (f) Loss of Quadrant 4

5. CONCLUSIONS

WALSHSTORE demonstrates that Walsh transform can be used in order to achieve a fail-soft storage of pictures and that Burst Processing is attractive in this application, especially in the front end, i.e., original picture storage and transformer. The use of Burst Processing in these parts of the system does not increase cost and helps reduce system hardware complexity due to simple encoder, processors and their controllers. The use of shift register storage reduces the system's IC count and the total number of chip interconnections needed for high reliability.

A WALSHSTORE type system can be used in various applications. In its current form it is suitable for still images since subpicture size is relatively high and processing is slow. However, reduction of subpicture size and use of processor arrays can increase computation speed significantly. For example, if a 256x256 picture is to be transformed using 16x16 subpictures, the transform period with a 10 Mhz clock signal is approximately 1.6 sec. using sequential subpicture processing. If parallel picture processing is incorporated, i.e., one processor assigned to one or a few subpictures, this time can be reduced. For the example above the extreme case is when 256 processors are used, one per subpicture, and the transform period is only 6.4 msec. Obviously, a compromise can be reached where the number of processors is minimized with, still, ON-LINE operation.

In WALSHSTORE all the transform coefficients are retained for fail-softness and its implementation attempts to use the simplest way to achieve this fail-softness. However, as discussed earlier, many coefficients are not important and may even be discarded. Using this property a different

fail-soft system can be designed which is more complex in hardware but is probably superior. In one such system the transformer tests each coefficient against a threshold and discards it if it is smaller than the threshold. For fail-softness the retained coefficients, which are very important now, must be duplicated or even triplicated. Another way can be to encode each coefficient with a different number of bits depending on its statistical variance as measured from previous pictures. Such systems can achieve high data compression ratios but their hardware complexity is much higher than the straight forward system described here.

Finally, a few words about the use of microprocessors. Since current microprocessors are slow and WALSHSTORE processors require a relatively high clock signal frequency, it is felt that their use is very limited in this application. Additionally, the processors have to perform only a small number of operations and, therefore, the microprocessor's flexibility is not required. However, if other operations, such as image enhancement, are to be performed, the microprocessor may be suitable. This argument applies also to the use of a microprocessor as a controller, i.e., if only one kind of operation is to be performed by the system it controls, a special purpose controller is much better, but if the system has to perform many operations the flexibility of the microprocessor makes it extremely useful.

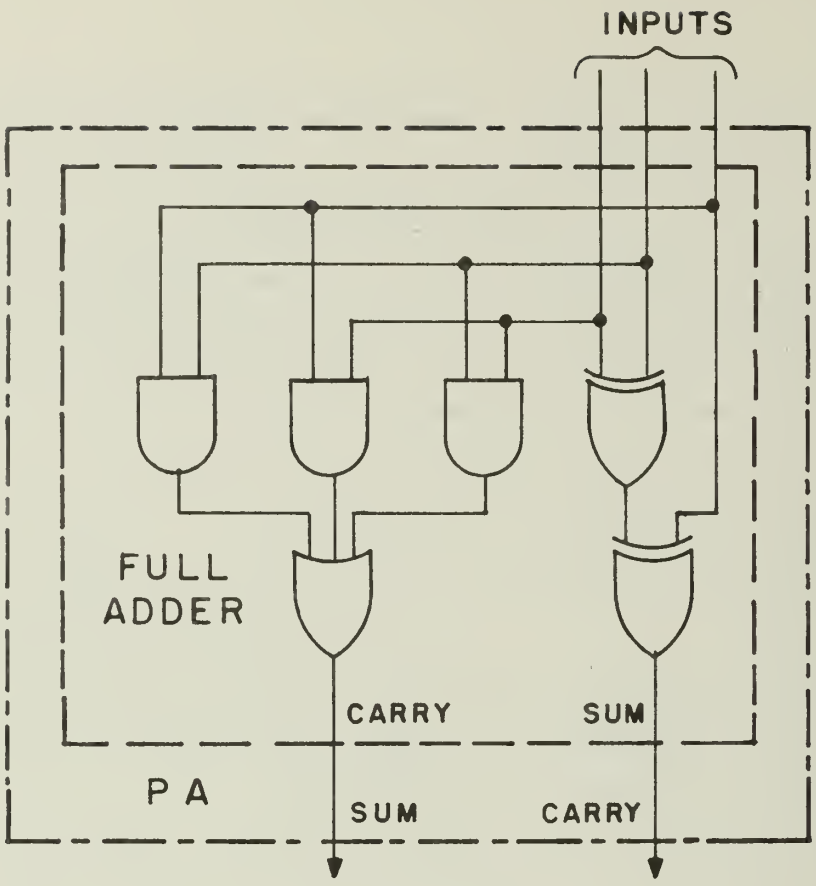
APPENDIX A

PA's and PA-Trees

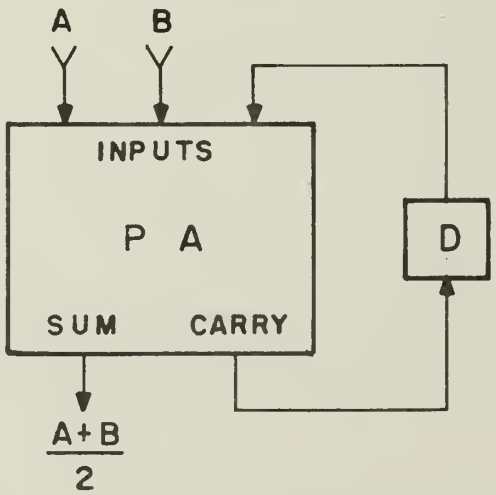
Figure A-1(a) shows the PA logic diagram. The PA is a full adder in which the carry is used as the sum output and the sum as the carry output. It is easy to show that the perverted sum output represents the sum of the three unary (Burst) inputs scaled by two. The perverted carry represents the remainder. For example, if the inputs are 1, 1, 0 the perverted sum is 1 and the perverted carry is 0, as expected. Figure A-1(b) shows a serial Burst adder using a PA and a carry FF. The carry output is added at the next cycle and is, therefore, not lost. This permits a long computation with a truncation only at the last addition.

When more than two Bursts are to be added, a PA-Tree can be used where each PA receives a pair of inputs. Since the inputs to each level of such a tree are of the same weight, their carry FF's are also of the same weight and they can be combined into one carry FF, as shown before in Figure 3.2. This reduces the number of FF's in the tree. In general, a tree with L levels scales by 2^L and may have a maximum of 2^L inputs. Such a tree requires 2^L-1 PA's and L carry FF's.

A PA-Tree can be used wherever parallel Burst addition is to be performed, either all the pulses in one Burst or one pulse from several Bursts.



(a)



(b)

Figure A-1 Perverted Adder Diagram

(a) Logic Diagram

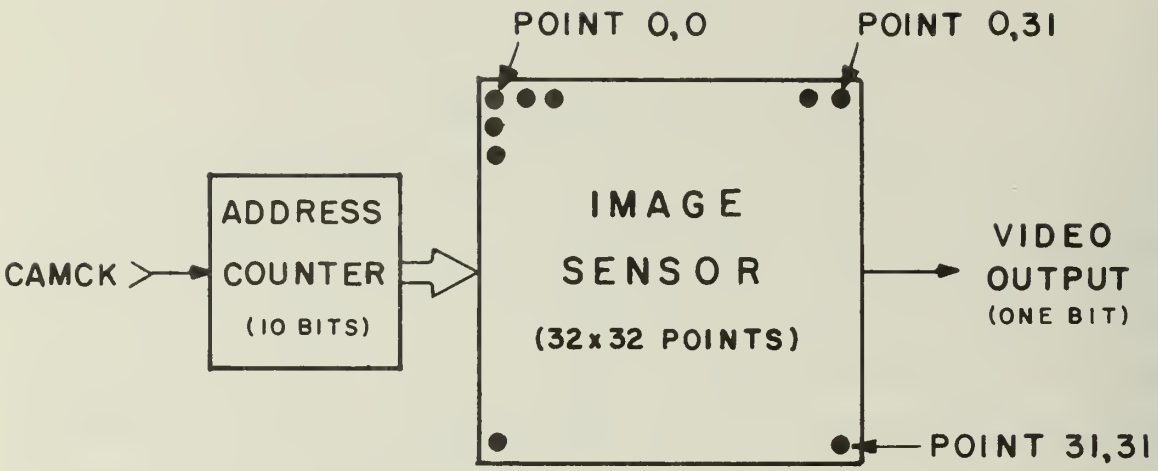
(b) Serial Burst Adder

APPENDIX B

The CYCLOPS Digital Camera

The CYCLOPS digital camera includes an image sensor addressed by a 10-bit counter used to define the point being accessed on the 32x32 array of points. Each time a point is accessed the sampled video output is compared to a threshold voltage, and then a 0 or a 1 is produced depending on the relation of the video voltage to the threshold. Every clock signal pulse increments the address counter, and the next point is so sampled. After each complete scan of the picture (field), the scan is repeated again. After 15 such fields the camera is reset and then during the next field all the image sensor points are cleared as a preparation for the next frame.

The operation of each image sensor point is such that during the 15 fields the light falling on the point is integrated and sampled at regular intervals by means of the address counter. If a point is completely dark the sampling produces only 0's. If it is completely bright the sampling produces 15 1's. For a partially lighted point the sampling produces the appropriate number of 1's. Figure B-1 shows the output video format for the CYCLOPS camera. Field 0 is used for reset and fields 1 to 15 are used for data outputs. The intensity of point 0,0 is also shown. Note that the intensity is Burst encoded except that the pulses for each point are distributed in time, and there is no need for any encoder for the video signal.



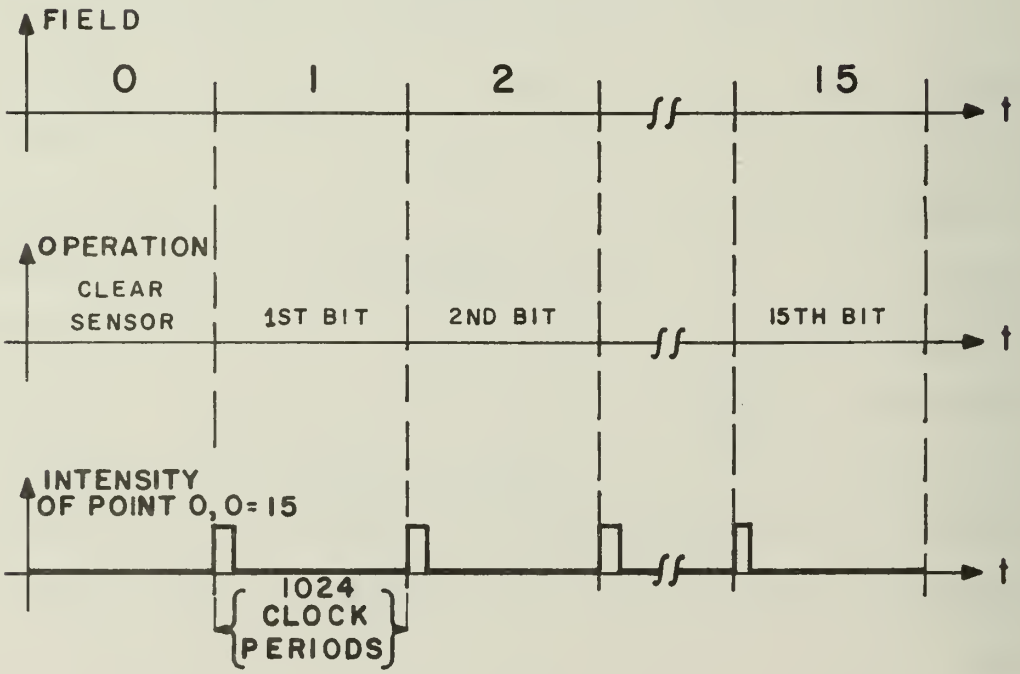


Figure B-1 The CYCLOPS Digital Camera Output Video Format

APPENDIX C

Operating Instructions for WALSHSTORE

Figure C-1 shows a photograph of WALSHSTORE's front panel with CYCLOPS and the IS and RIS display units. The following steps should be followed for proper operation of the machine:

1) Preparations and Power-On

Connect the camera cable to both WALSHSTORE and the camera. Connect the 6 coaxial cables for X, Y, and Z signals to the display units. Set the intensity polarity switches, located on cards 3 and 15, to the appropriate polarity. Set all the front panel toggle switches to the up position. Plug WALSHSTORE's and both display units' AC cords into a 115V 60 Hz outlet and switch them on. (Note that WALSHSTORE's power switch is located on the AC cord.) The machine automatically enters the IS Display and RIS Display modes at power-on.

2) Input of a Picture

Depress and release the INPUT push-button to transfer to the Input mode. Aim the camera at the picture and observe the picture actually being picked up, ON-LINE, on the IS display unit. When the desired picture is displayed, depress and release the INPUT push-button to exit the Input mode and to return to the IS Display mode. The picture is now stored in IS and also displayed on the IS display unit. (Note that the camera may be moved now since the picture is frozen in memory.)

3) Transform and Inverse Transform

To transform the picture, depress and release the TRANSFORM push-button. The machine goes to the Transform mode and returns to the IS Display mode

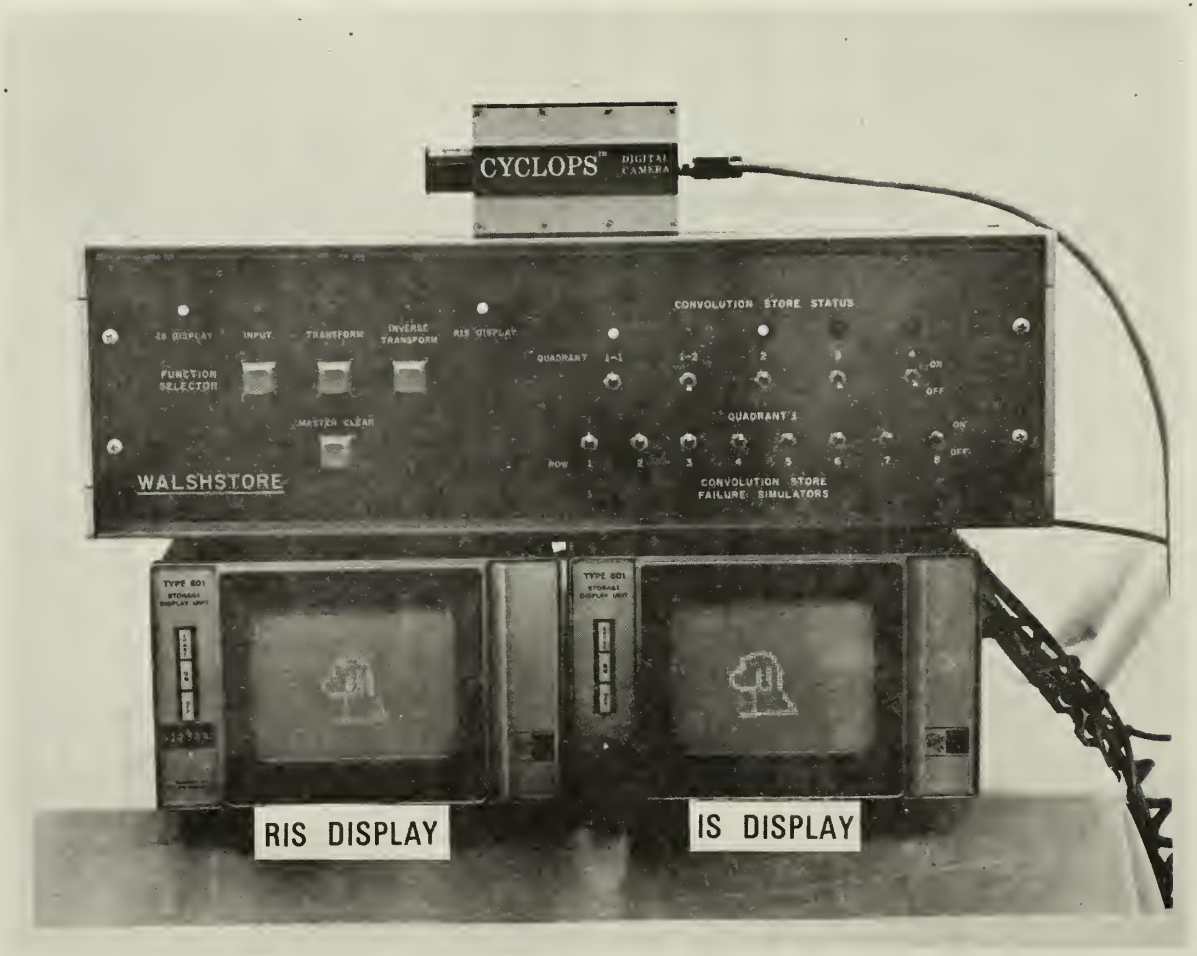


Figure C-1 Photograph of WALSHSTORE with CYCLOPS and Display Units

after 2 sec. approximately. During this period the IS display unit is blanked.

To inverse transform a transformed picture, depress and release the INVERSE TRANSFORM push-button. The machine goes to the Inverse Transform mode and returns to the RIS Display mode after 0.5 sec. approximately, and during this period the RIS display unit is blanked.

4) Loss Simulation

Once a transform is performed, loss simulation can be performed by switching off full quadrants or individual rows in the first quadrant by means of the CS failure simulator switches on the front panel. An inverse transform must be performed in order to observe the effects of the losses on the inverse transformed picture.

IMPORTANT:

Every time the full quadrant switch settings are changed a transform must be performed, since data is lost whenever a quadrant is switched off. However, the loss of rows in the first quadrant is not permanent and, therefore, does not require a transform for each new setting for these switches.

LIST OF REFERENCES

[1] Rosenfeld, A. and Kak, A. C., Digital Picture Processing, Academic Press, New York, 1976.

[2] Wolff, M., "Transmission of Analog Signals Using Burst Techniques," Department of Computer Science, University of Illinois, Urbana, Illinois, January 1977.

[3] Taylor, G. L., "An Analysis of Burst Encoding Methods and Transmission Properties," Department of Computer Science, University of Illinois, Urbana, Illinois, December 1975.

[4] Proceedings, IEEE, (A Special Issue on Image Processing) July 1972

[5] Poppelbaum, W. J., Proposal to ONR entitled "Application of Stochastic and Burst Processing to Communication and Computing Systems," Department of Computer Science, University of Illinois, Urbana, Illinois, July 1975.

[6] Ahmed, N. and Rao, K. R., Orthogonal Transforms for Digital Signal Processing, Springer-Verlag, New York, 1975.

[7] Harmuth, H. F., Transmission of Information by Orthogonal Functions, Second Edition, Springer-Verlag, New York, 1972.

[8] Clarke, C. K. P. and Walker, R., "Walsh-Hadamard Transformation of Television Pictures," Proceedings of Conference on Applications of Walsh Functions, 1974.

[9] Pleva, R. M., "A Microprocessor-Controlled Interface for Burst Processing," Department of Computer Science, University of Illinois, Urbana, Illinois, July 1976.

[10] Mohan, P. L., "The Application of Burst Processing to Digital FM Receivers," Department of Computer Science, University of Illinois, Urbana, Illinois, January 1976.

[11] Bracha, E., "BURSTCALC (A BURST CALCulator)," Department of Computer Science, University of Illinois, Urbana, Illinois, October 1975.

[12] Xydes, C. J., "Application of Burst Processing to the Spectral Decomposition of Speech," Department of Computer Science, University of Illinois, Urbana, Illinois, July 1977.

[13] Wells, D. K., "Digital Filtering Using Burst Processing Techniques," Department of Computer Science, University of Illinois, Urbana, Illinois, July 1977.

[14] Wintz, P. A., "Transform Picture Coding," Proceedings, IEEE, July 1972

VITA

Ehud Bracha was born on October 9, 1947 in Tiberia, Israel. He received his B.Sc. in Electrical Engineering Cum Laude from the Technion, Israel Institute of Technology, Israel in June 1973 and his M.S. and Ph.D. in Computer Science from the University of Illinois in January 1976 and January 1978, respectively.

Since August 1974 he has been a research assistant in the Information Engineering Laboratory in the Department of Computer Science at the University of Illinois, under the direction of Professor W. J. Poppelbaum.

He is a member of IEEE and IEEE Computer Society.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: UIUCDCS-R-77-878	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: WALSHSTORE: THE APPLICATION OF BURST PROCESSING TO FAIL-SOFT STORAGE SYSTEMS USING WALSH TRANSFORMS	5. TYPE OF REPORT & PERIOD COVERED: Ph.D. Thesis, Oct., 1977
	6. PERFORMING ORG. REPORT NUMBER: UIUCDCS-R-77-878
7. AUTHOR(s): Ehud Bracha	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0982
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science University of Illinois Urbana, IL 61801	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research Code 437 Arlington, Virginia 22217	12. REPORT DATE: Oct., 1977 13. NUMBER OF PAGES: 69
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Release Unlimited 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Burst Processing	Fail-Soft Storage
Walsh Functions	Data Compression
Walsh Transform	Orthogonal Transform

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

WALSHSTORE is concerned with the investigation into the application of orthogonal transforms, e.g., Walsh transforms, to the storage of visual images in a fail-soft manner, i.e., the system can sustain losses but the retrieved image can still be recognized, even if in a degraded form.

The application of Burst Processing to various parts in such a system is investigated together with an analysis of the amount of storage and speed

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20.

of computation that can be achieved.

This paper presents the necessary theory and the implementation of the Walsh transfomers and the fail-soft storage which were constructed and tested. Additionally, some software simulation results are compared to the actual machine performance.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-77-878	2.	3. Recipient's Accession No.

4. Title and Subtitle

WALSHSTORE: THE APPLICATION OF BURST PROCESSING TO FAIL-SOFT STORAGE SYSTEMS USING WALSH TRANSFORMS

5. Report Date: October, 1977

6.

7. Author(s)

Ehud Bracha

8. Performing Organization Rept. No. UIUCDCS-R-77-787

9. Performing Organization Name and Address

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

N00014-75-C-0982

12. Sponsoring Organization Name and Address

Office of Naval Research
Code 437
Arlington, VA 22217

13. Type of Report & Period Covered

Ph.D. Thesis

14.

15. Supplementary Notes

16. Abstracts

WALSHSTORE is concerned with the investigation into the application of orthogonal transforms, e.g., Walsh transforms, to the storage of visual images in a fail-soft manner, i.e., the system can sustain losses but the retrieved image can still be recognized, even if in a degraded form.

The application of Burst Processing to various parts in such a system is investigated together with an analysis of the amount of storage and speed of computation that can be achieved.

This paper presents the necessary theory and the implementation of the Walsh transformers and the fail-soft storage which were constructed and tested. Additionally, some software simulation results are compared to the actual machine performance.

17. Key Words and Document Analysis. 17a. Descriptors

Burst Processing
Walsh Functions
Walsh Transform
Fail-Soft Storage
Data Compression
Orthogonal Transform

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement

Release Unlimited

19. Security Class (This Report) UNCLASSIFIED

20. Security Class (This Page) UNCLASSIFIED

21. No. of Pages 69

22. Price

JAN 25 1978

UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R no. C002 no.874-879(1977
INDUCE-1 : an interective inductive infe
3 0112 088403453